GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

PROJECT ADMINISTRATION DATA SHEET

[X] ORIGINAL [ ] REVISION NO. _____

Project No. (R6122-0A0) E-21-628 GTRC~~XXX~~ DATE 5 / 7 / 86

Project Director: Dr. George Vachtsevanos School/~~XXX~~ EE

Sponsor: Georgia Power Company

Type Agreement: Purchase Order No. D-52012

Award Period: From 4/7/86 To ~~10/6/86~~ 12-31-86 4-30-87 (Performance) 10/6/86 (Reports)

Sponsor Amount:	This Change	Total to Date
Estimated:	$ 20,000	$ 20,000
Funded:	$ 20,000	$ 20,000

Cost Sharing Amount: $ None Cost Sharing No: N/A

Title: Investigation Potential Islanding of Dispersed PV Systems

ADMINISTRATIVE DATA OCA Contact William F. Brown X4820

1) Sponsor Technical Contact:

Mr. E. J. Ney, Mgr.
Georgia Power Company-Solar Operations
7 Solar Circle
Shenandoah, GA 30265
(404) 253-0218
Atlanta Line (404) 526-4745

2) Sponsor Admin/Contractual Matters:

fense Priority Rating: N/A Military Security Classification: N/A
(or) Company/Industrial Proprietary: N/A

STRICTIONS

Attached N/A Supplemental Information Sheet for Additional Requirements.

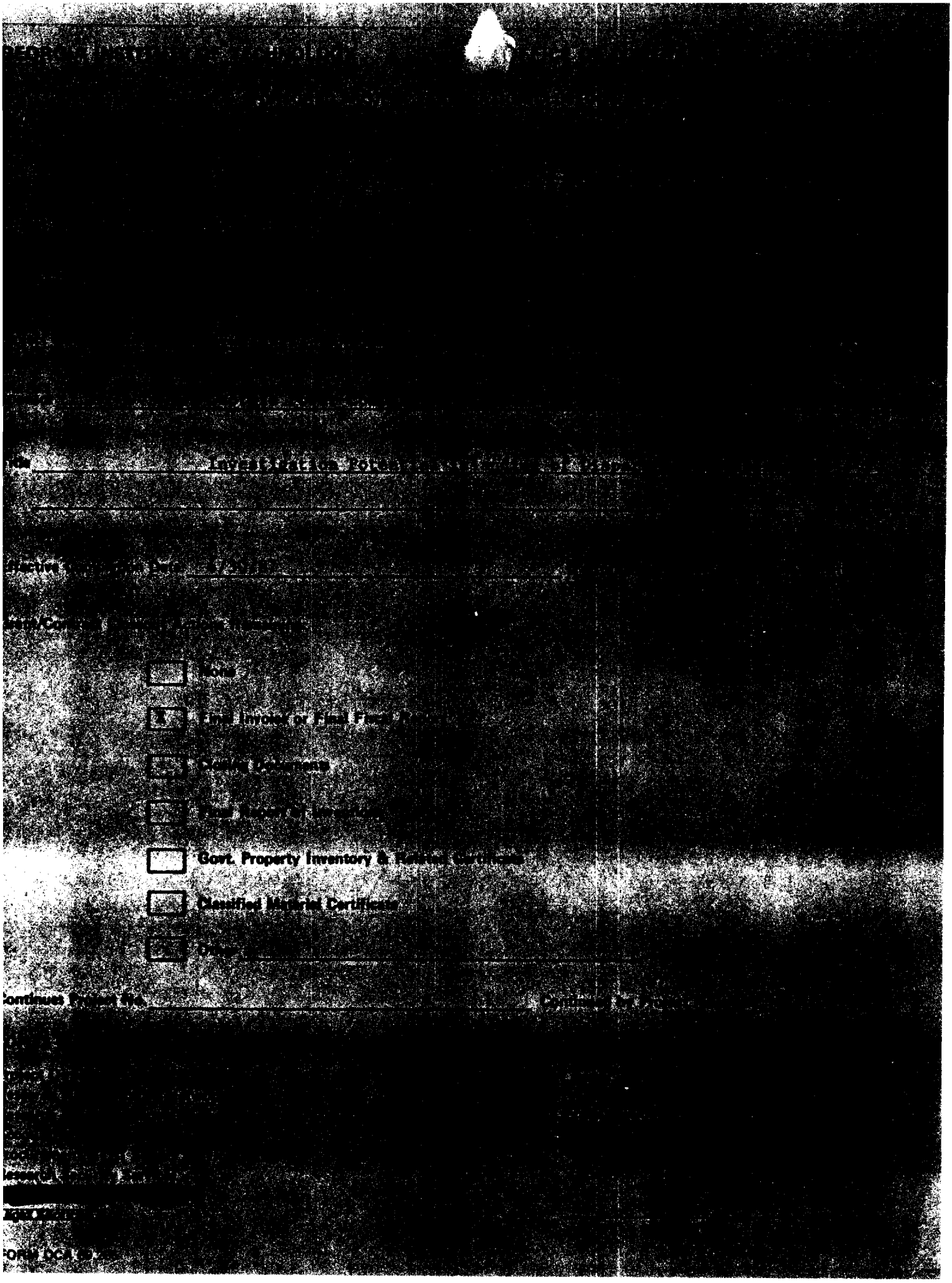
vel: Foreign travel must have prior approval – Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

ipment: Title vests with None proposed or anticipated

MMENTS:

ES TO: SPONSOR'S I. D. NO. 02.256.000.86.021

ct Director	Procurement/GTRI Supply Services	GTRC
arch Administrative Network	Research Security Services	Library
arch Property Management	Reports Coordinator (OCA)	Project File
unting	Research Communications (2)	Other A. Jones/Legal



E-21-628

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING
ATLANTA, GEORGIA 30332

LEPHONE: (404) 894- 2961

September 18, 1986

MEMORANDUM

TO: Ms. Pat Heitmuller
Printing and Photographic Center

FROM: Cindy Meyer
School of Electrical Engineering

SUBJECT: E21-628 Deliverables

For the months of May, July, and August, 1986, brief oral reports were given to the technical monitors of this contract at Georgia Power Company.

G. Vachtsevanos, Project Director

CM

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING
ATLANTA, GEORGIA 30332

PHONE (404) 894-

May 21, 1986

Mr. Dennis Keebaugh
Georgia Power Company-Solar Operations
7 Solar Circle
Shenandoah, Georgia 30265

Dear Mr. Keebaugh:

Enclosed is a copy of the first monthly report, dated May 14, 1986, for project No. E-21-628 "Investigation of Potential Islanding of Dispersed PV Systems".

Sincerely,

George Vachtsevanos
Professor
School of Electrical Engineering

GV/klm

Enclosure

May 14, 1986

Monthly Report

April 1986

Investigation of Potential Islanding of Dispersed Photovoltaic Systems

. Attended the project kick-off meeting on February 20 in Birmingham.

. Received from Bob Jones on March 12 data pertaining to TVA's PV houses and a list of initial islanding simulations to be conducted with the Teslaco SPC model. I reviewed the proposed simulation cases and I transmitted my comments by telephone to Bob Jones. In summary, for case A, I suggested that the resistive load be made continuously variable $\pm$ 10% from the matched condition. For case B, the feeder impedance must be modelled carefully to reflect the presence of both inductive and capacitive loading. Resonances along the distribution feeder will impact significantly islanding. The concept of the islanding "window" was revisited in terms of

(a) deviation from matched generation/load conditions ($\pm$ %)
(b) duration as compared to maximum relay reclosure time (typically 22 cyles)

These "windows" should be used as a benchmark for both the analytical and the experimental studies. It was suggested that the TVA feeder should be used for these initial simulations to avoid duplication and gain some insight into the behavior of the test system.

For cases C, D and E, attention must be focused at the "inertia" of the islanding part of the system. Particularly, the effect of voltage/current and frequency variations on rotating equipment must be modelled accurately. Rotating masses may change their operating mode (from motor to generator or vice versa) under varying excitation conditions. In turn, this varied behavior will impact on the islanding phenomenon itself. Nonlinearities of such devices as transformers, motors, etc. will also need to be represented to account for harmonic/subharmonic phonomena.

The proposed simulations with multiple Teslaco units are a necessary component of the analytical work and are well planned.

I received a copy of the Purdue submission to SCS on the Teslaco static power converter schematic and the analysis procedure (4/14). I discussed the contents via telephone with Bob Jones. The control system block diagram did not include the ac overcurrent and dc under/over voltage controls. Under normal operating conditions these protective devices will most likely be inactive. Under islanding conditions though, these devices are an integral part of the control/protection circuitry and should be included in the simulation studies. In the event that they are not, then verification of the analytical results with experimental data might not be always possible. Some refinements of the system block diagram might be in order later on to account for nonlinearities if there is no agreement between analytical and experimental results.

The development of a simplified functional model for the DECC or APCC static power converters is being delayed due to two factors. Firstly, there is no detailed schematic information available on either one of these two inverters. Efforts are currently under way through SANDIA to obtain such information. Secondly, the present phase of the Teslaco simulations must be completed so that simplifications and generic features may be extracted from these studies that will be useful in developing a DECC or APCC model.

E-21-621

July 7, 1986

Monthly Report

June 1986

On June 5 and 12, 1986, I met with Drs. Oleg Washynczuk and Paul Krausse at Purdue University, West Lafayette, Indiana and discussed the hybrid model development for the Teslaco inverter as well as the preliminary information made available to Purdue by American Power Conversion Corporation on their inverter.

On the basis of the material provided to me by Drs. Washynczuk and Krausse, a preliminary simplified analytical model of the Teslaco inverter is being developed and a brief description of progress to date is appended to this report.

Furthermore, requirements have been specified for the measurement program to be performed by GP during the experimental phase of the project. These data will be used to tune and verify, to the maximum degree possible, the simplified model.

As more detailed information on the APCC inverter model becomes available from Purdue, attention will be focused on the development of a similar simplified analytical model for this type of inverter.

A Simplified Analytical Model for the Teslaco 4 Kw Inverter

The buck stage of the inverter is represented first as a Thevenin equivalent circuit. Figure 1 shows the block diagram of the buck stage current feedback loop as provided by Purdue and developed in Reference 1.

From Figure 1:

$$V_c = I_{ref} - \beta I_{out}$$

$$V_{out} = \alpha V_c = \alpha I_{ref} - \alpha\beta I_{out}$$

The Thevenin dynamic resistance is defined as

$$R_{th} = \frac{dVout}{dIout} = \alpha\beta = 6.24\ \Omega$$

and it is a nondissipative synthesized element.

It is seen that

$$V_s = \alpha I_{ref} = 390 I_{ref}$$

Figure 2 is the Thevenin equivalent circuit representation of the buck stage.

Figure 3 shows the unfolding stage, noise filters, residential load and the utility equivalent representation loading the inverter buck stage. I_{ref} is a FWR sinusoidal signal whose amplitude is modulated by a slowly-varying I_{mag} and whose frequency is appropriately varied by the Phase Locked Loop (PLL) circuitry. It is assumed that solar insolation levels remain fixed and, therefore, the value of I_{mag} will be taken as a constant. Actually, V_s may be expressed as

$$V_s(t) = 390 I_{mag}(t) \sin 2\pi [f(t)] t$$

Next, the effect of the discontinuous nature of the FWR waveform is neglected and, since the gain of the unfolder stage is unity, V_s is represented as a pure sinusoid.

As a final result, grouping the peripheral elements together, the simplified inverter equivalent circuit may be drawn as shown in Figure 4.

The key element in representing the dynamic behavior of the inverter under "islanding" or isolation conditions is the PLL circuitry which incorporates suitable devices to detect phase differences between the utility line and the inverter equivalent voltages. Upon detection of a phase difference greater than approximately 6°, the inverter transistor bridge is disconnected from the load. This is accomplished by incorporating in the PLL a 1.00 cycle time delay which destabilizes the loop upon line disconnection.

Refer to Figure 5 showing a block diagram of the PLL circuitry (from Reference 1). The output of the loop filter (V_y) is zero whenever the frequency is exactly 60 Hz and the phase comparator is locked at 0°.

The output of the XOR gate is low in this case, since its two inputs are exactly in phase. When the frequency changes, the time delay is no longer exactly one cycle and the two inputs to the comparator are phase shifted.

The loop filter output, V_y produces a frequency deviation of 920 Hz for every volt increase (or decrease) in V_y The RC filter after the XOR gate is designed so that the Schmitt trigger trips and disconnects the inverter whenever the phase error is greater than about 6° (corresponding to a 1.00 Hz frequency deviation). It is important, therefore, to model analytically V_y as a function of inverter, load and feeder conditions. Test data indicate that should be modeled as a step followed by an expoential growth. No detailed modeling of V_y is required (intracyclic variations) since the RC filter acts as an integrator actuating the transistor disconnect sequence on the basis of average V_y data. An accurate representation of V_y will assist in determining a reasonable model of the frequency modulated Thevenin voltage. From that point on, the time required for the frequency deviation to reach 1.00 Hz may be determined.

When the load is purely resistive and, therefore, there is no phase shift then the trigger mechanism is actuated by unbalances in the magnitude of the voltage. Ideally, an inverter may operate in an isolated mode if the load is purely resistive and matches exactly the PV generation. Such a condition cannot be achieved under normal operation and inherent noise conditions will tend to destabilize the PLL circuit.

Preliminary information from Purdue indicates that the APCC inverter is designed to operate very similarly to the Teslaco inverter. The same destabilizing philosophy is employed for this device. Some basic differences refer to

- use of a XOR phase detector
- there is no time delay of 1.00 Hz in the PLL circuit
- a resonant line filter (0° at 60 Hz) is used to provide for the destabilizing action

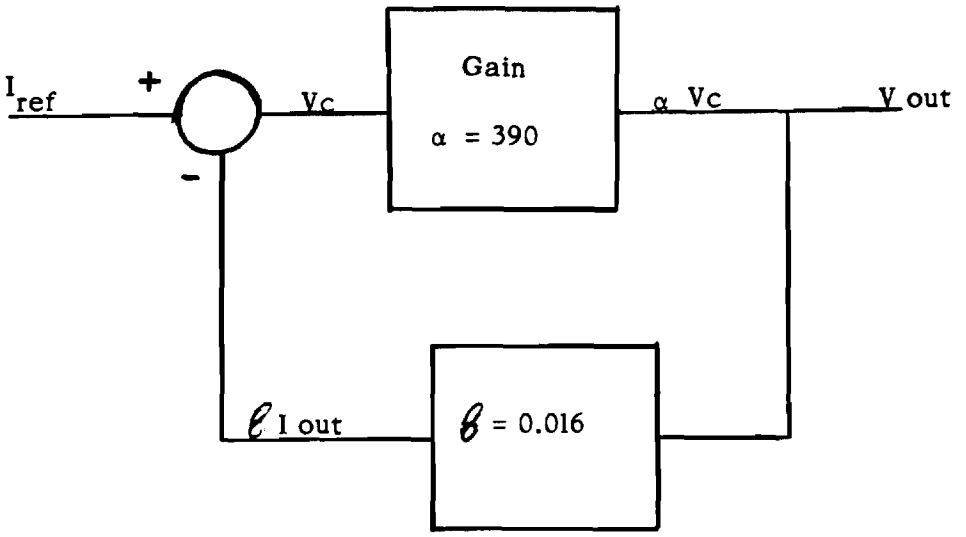


Figure 1. Buck Stage Current Feedback

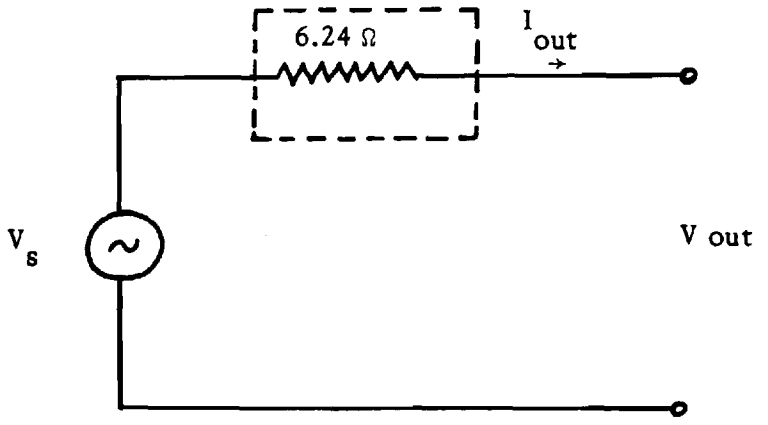


Figure 2. Buck Stage Thevenin Equivalent Circuit

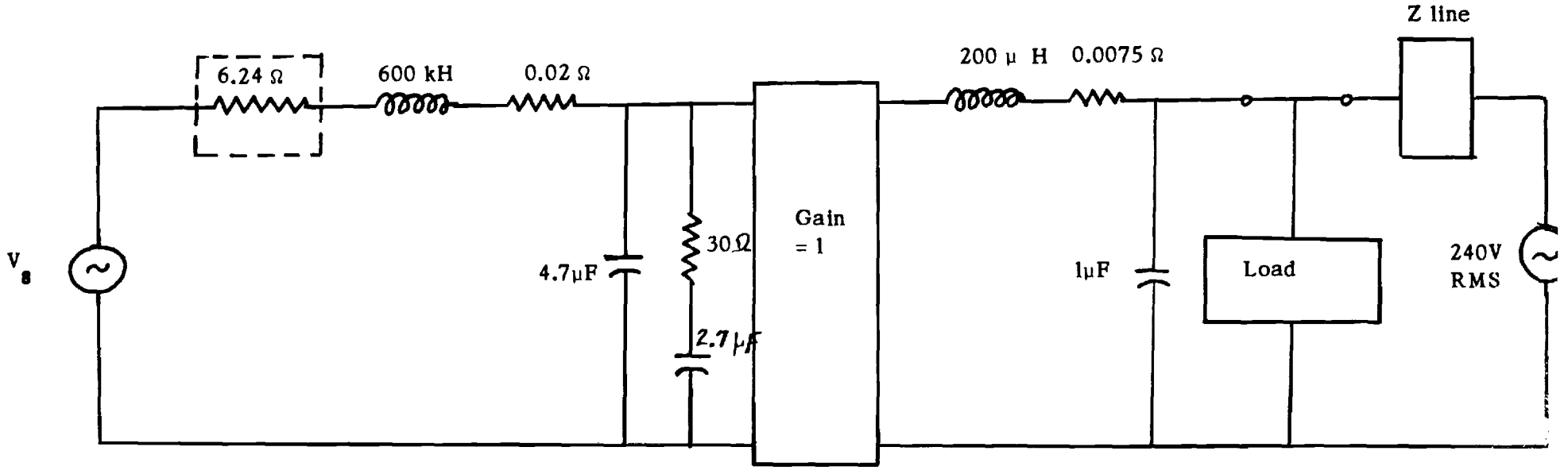


Figure 3. Equivalent Circuit Representation of the Inverter

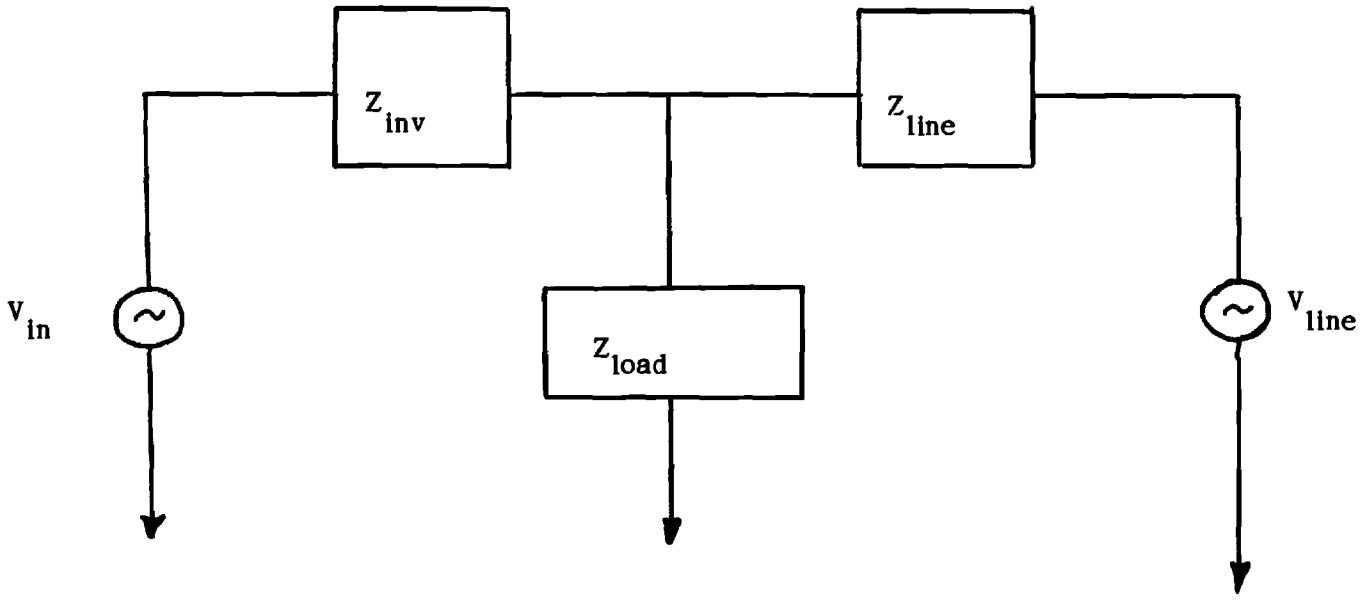


Figure 4. Inverter - Load - Utility Line Equivalent Circuit

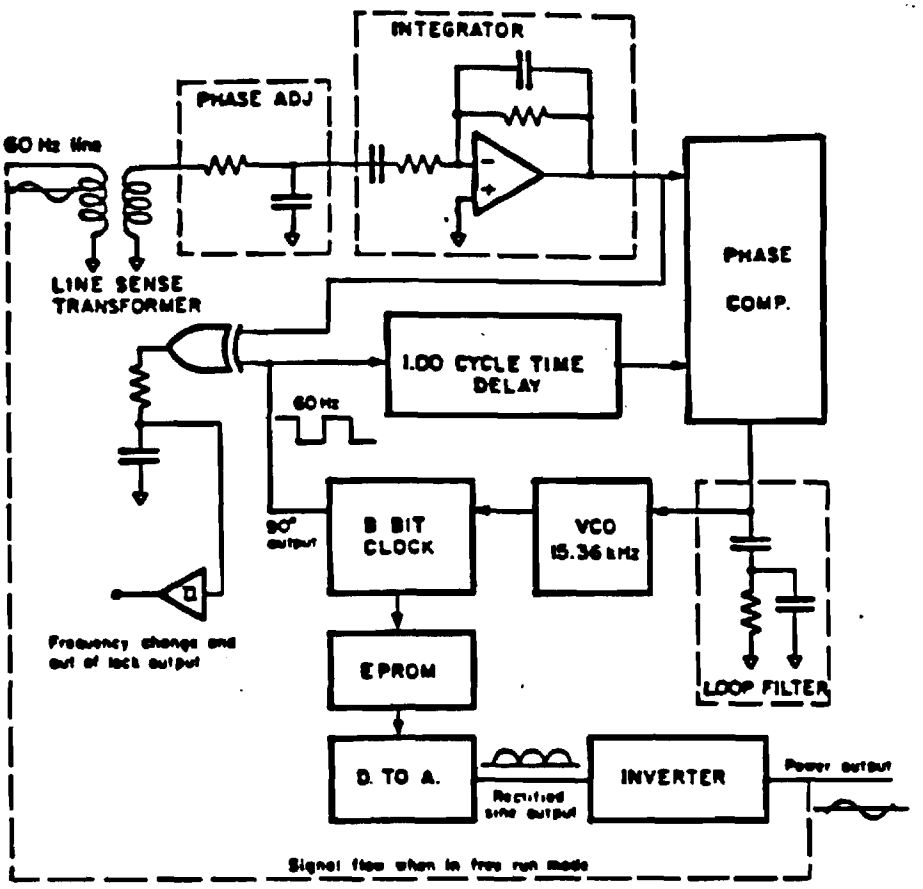


Figure 5. Phase Locked Loop Circuit

December 1986

Monthly Report

Project: Investigation of Potential Islanding of Dispersed PV Systems

Georgia Tech has been developing a simplified model for the TESLACO inverter in anticipation of design data for the APCC inverter from Purdue University through SANDIA.

They have modeled the PLL of the inverter and ran several simulations with reasonable and encouraging results. In addition, the buck state and the unfolder state have been modeled. Various load conditions are simulated. The progam is written in GW-BASIC and runs on the IBM XT personal computer. A program listing and some preliminary results are enclosed with this report.

On the Operation of the PLL

The main function of the line-locked PLL circuitry of the inverter deals with the effective out-of-phase or in-phase voltage between the utility line and the output of the inverter when the utility line is disconnected. This out-of-phase condition depends on the load characteristics, such as resistive and/or reactive loads. In addition, PLL generates a sinc wave reference voltage in phase with the utility line. The inverter has an ability of bidirectional power flow.

If the load is resistive, there is no phase shift upon the line disconnection. Now if the array power is changed, the output voltage of the inverter will be varying. If the load is reactive, or with non-unity power factor, the output voltage of the inverter begins to shift according to the phase from the inverter's internal voltage source. After all, this phase discrepancy is detected by the line locked PLL, increasing its output of the loop filter and finally resulting in the inverter shutdown.

In the steady state condition, the output of the loop filter in the PLL circuitry will be zero, making the PLL stable. As the phase shift occurs after the disconnection of the utility line, the output of the loop filter becomes unstable in such a way that for every volt-increment a frequency deviation of 920HZ results and if the output of the VCO exceeds 1.00 HZ frequency deviation or 6° phase error, the inverter system shuts down.

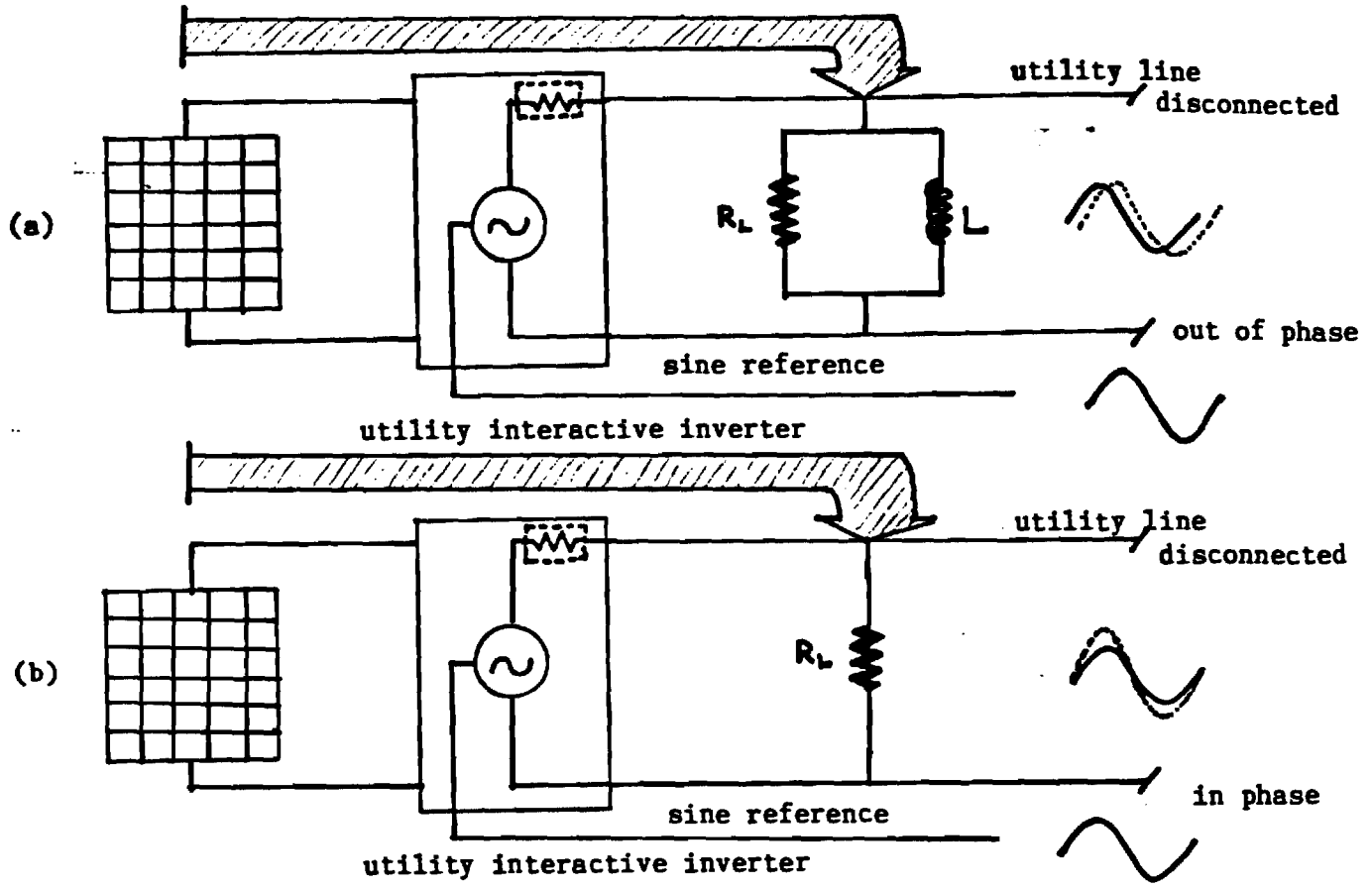


Figure. Effect of utility line disconnection

(a) finite phase shift for non-unity power factor load

(b) zero phase shift for unity power factor load

2. On the Simulations

The program using GW-BASIC is divided into two parts; one for the PLL and the SPC TESLACO INVERTER, and the other for the lineload and the user load.

In the present situation, we use the phase-shifted versions of two steady-state Vac voltages, i.e., Vac with line-connection and with line-disconnected. The next step is to find the run-on times of the SPC in the general load environments.

Figure 1: Vac, off = Vac, on $\angle + 0.3°$

Figure 2: Vac, off = Vac, on $\angle + 0.3° \left(\frac{t-Toff}{Ts}\right) * 0.05$

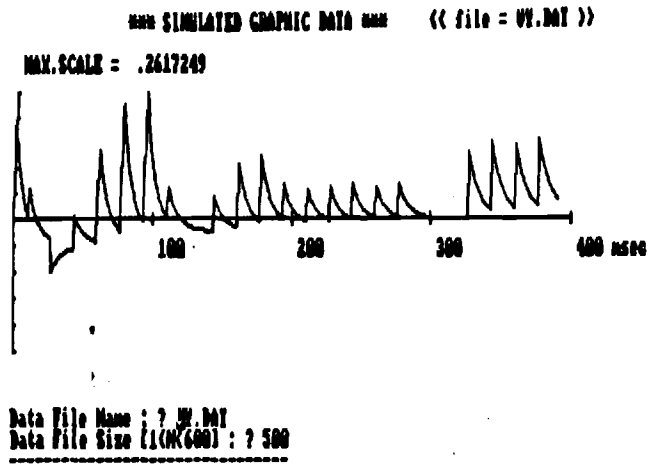
Figure 3: Vac, off = Vac, on $\angle + 0.3° \left(\frac{t-Toff}{Ts}\right) * 0.1$

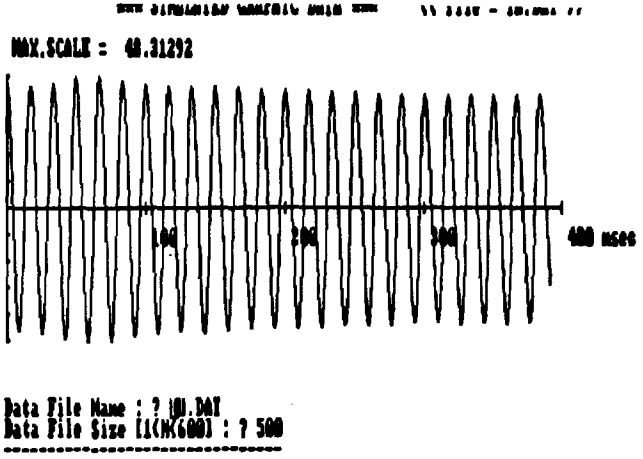
Figure 4: Vac, off = Vac, on $\angle + 0.3° \left(\frac{t-Toff}{Ts}\right) * 0.2$

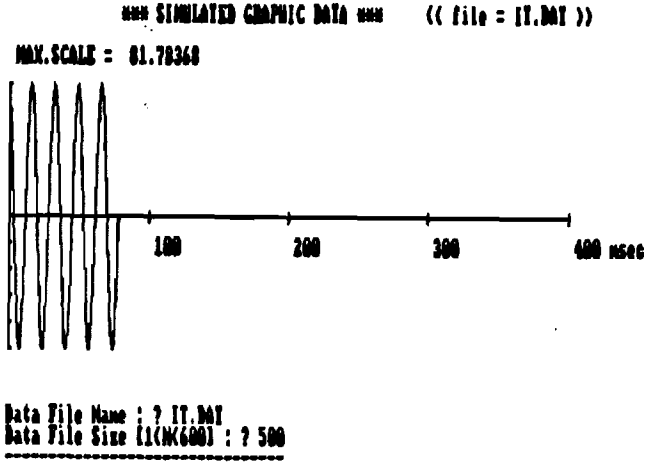
Figure 5: Vac, off = Vac, on $\angle + 0.3° \left(\frac{t-Toff}{Ts}\right) * 1$

Ts = sampling period (65.1 μ sec)

Toff = line-disconnection time (at 260 m sec)


=== SIMULATED GRAPHIC DATA ===
((file = VV.DAT))
MAX.SCALE = .2617249
100
200
300
400 msec
Data File Name : ? VV.DAT
Data File Size [1(MK600] : ? 500


MAX.SCALE = 48.31292
100
200
300
400 msec
Data File Name : ? UU.DAT
Data File Size [1(MK600] : ? 500


=== SIMULATED GRAPHIC DATA ===
((file = IT.DAT))
MAX.SCALE = 81.78368
100
200
300
400 msec
Data File Name : ? IT.DAT
Data File Size [1(MK600] : ? 500

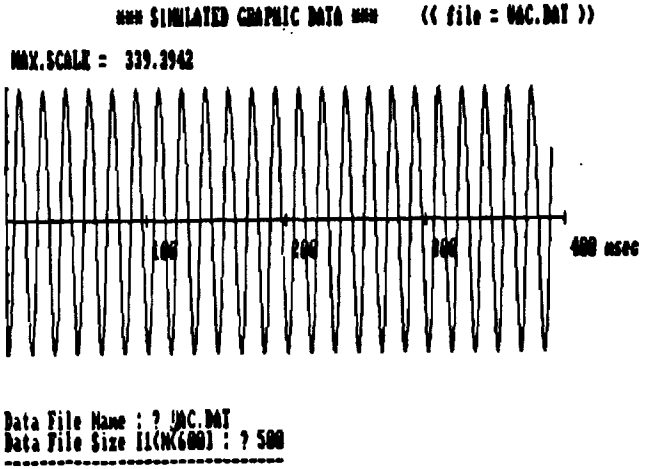
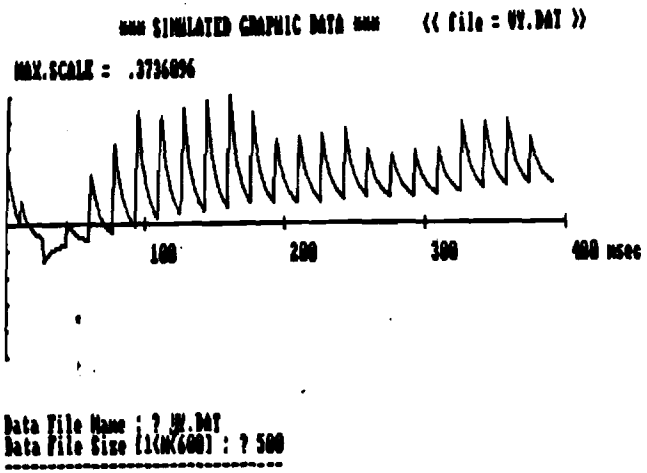
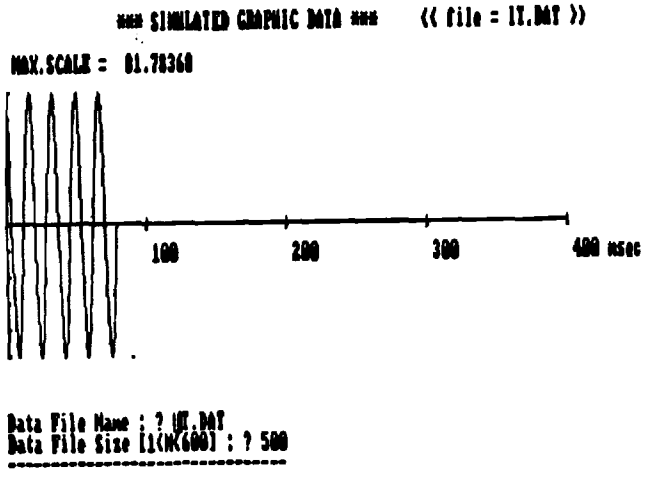
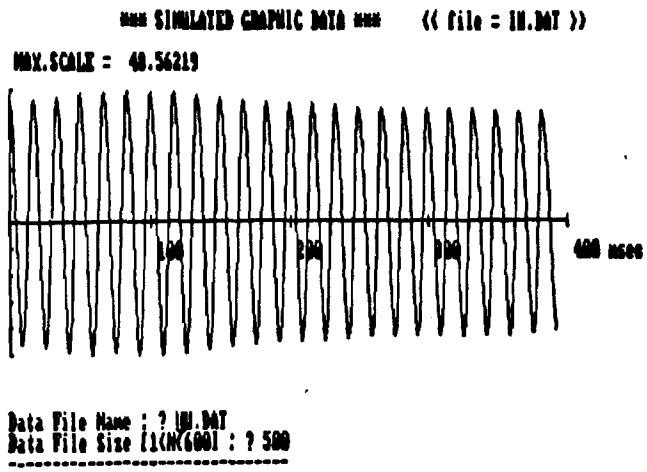

=== SIMULATED GRAPHIC DATA ===
((file = UAC.DAT))
MAX.SCALE = 339.3942
100
200
300
400 msec
Data File Name : ? UAC.DAT
Data File Size [1(MK600] : ? 500

Fig. 1.


*** SIMULATED GRAPHIC DATA ***
((file = VV.DAT))
MAX.SCALE = .3736896
100
200
300
400 msec
Data File Name : ? VV.DAT
Data File Size [1(M)600] : ? 500


*** SIMULATED GRAPHIC DATA ***
((file = IT.DAT))
MAX.SCALE = 81.78360
100
200
300
400 msec
Data File Name : ? IT.DAT
Data File Size [1(M)600] : ? 500


*** SIMULATED GRAPHIC DATA ***
((file = IN.DAT))
MAX.SCALE = 40.56219
100
200
300
400 msec
Data File Name : ? IN.DAT
Data File Size [1(M)600] : ? 500

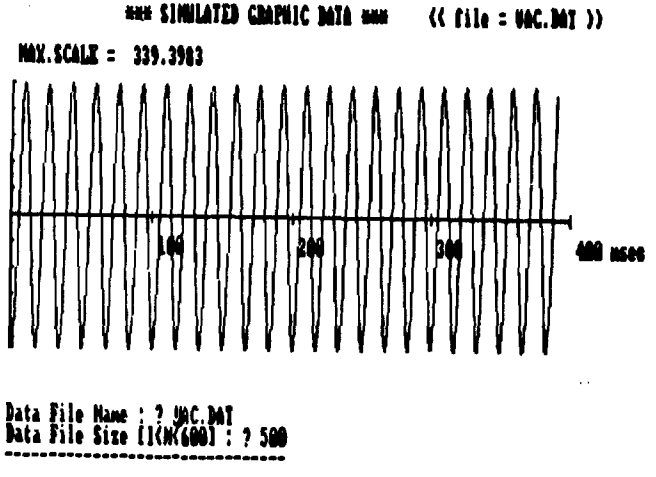
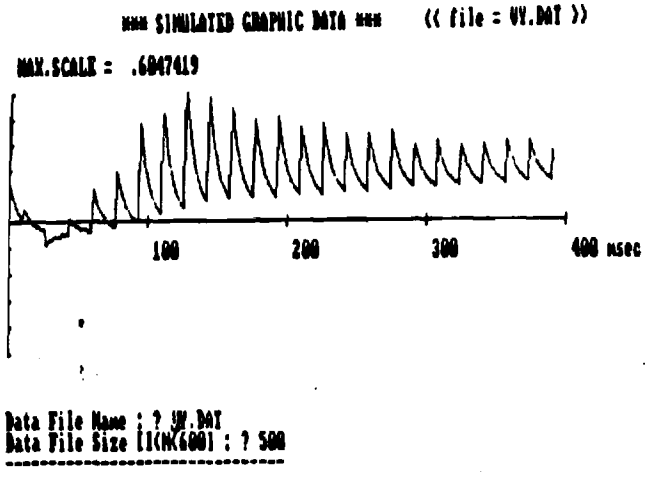
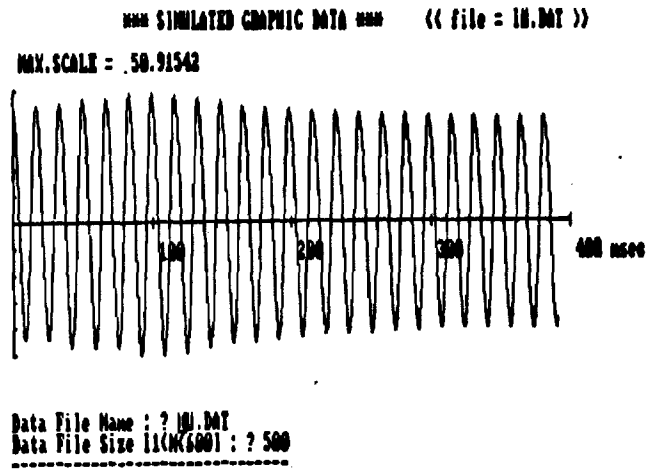
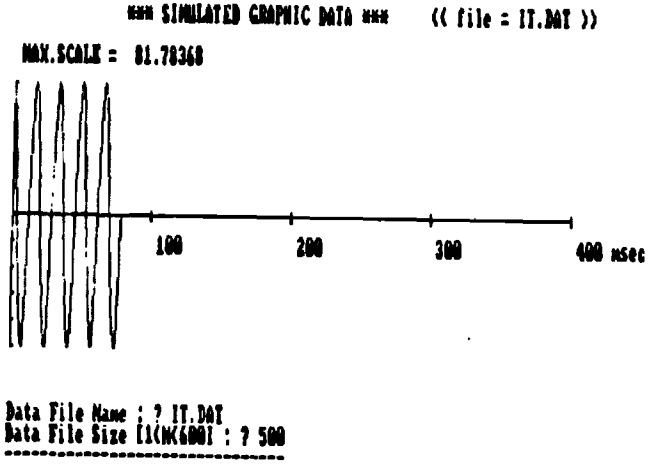

*** SIMULATED GRAPHIC DATA ***
((file = VAC.DAT))
MAX.SCALE = 339.3983
100
200
300
400 msec
Data File Name : ? VAC.DAT
Data File Size [1(M)600] : ? 500

Fig. 2.


=== SIMULATED GRAPHIC DATA ===
((file = VV.DAT))
MAX.SCALE = .6047419
100
200
300
400 msec
Data File Name : ? VV.DAT
Data File Size [1(NG600] : ? 500


=== SIMULATED GRAPHIC DATA ===
((file = IR.DAT))
MAX.SCALE = 50.91542
100
200
300
400 msec
Data File Name : ? IR.DAT
Data File Size [1(NG600] : ? 500


=== SIMULATED GRAPHIC DATA ===
((file = IT.DAT))
MAX.SCALE = 81.78368
100
200
300
400 msec
Data File Name : ? IT.DAT
Data File Size [1(NG600] : ? 500

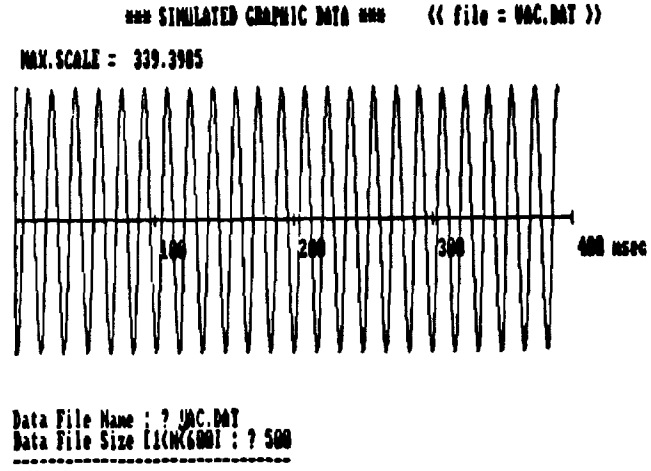
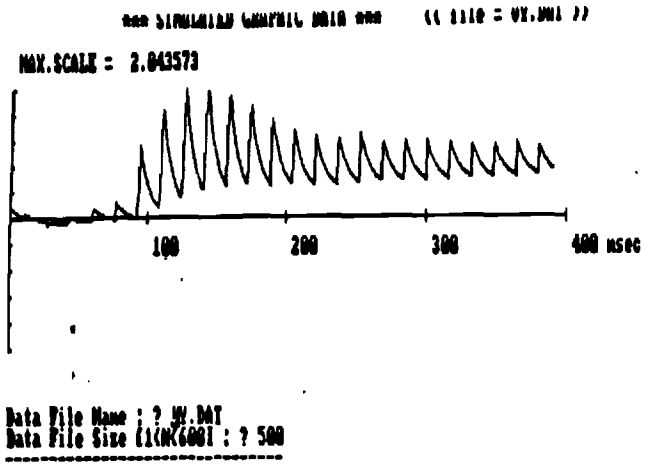
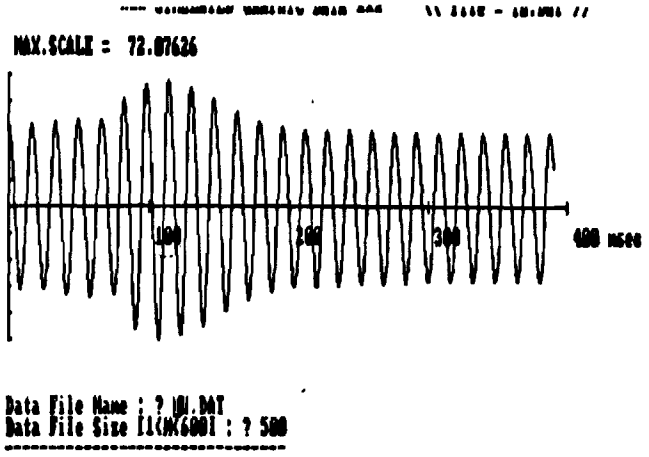
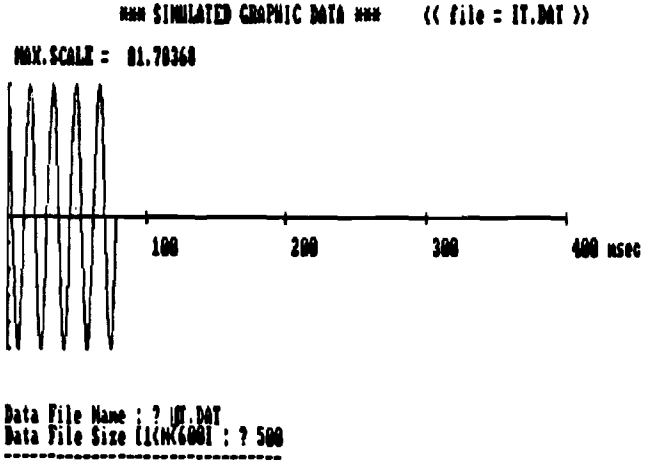

=== SIMULATED GRAPHIC DATA ===
((file = VAC.DAT))
MAX.SCALE = 339.3985
100
200
300
400 msec
Data File Name : ? VAC.DAT
Data File Size [1(NG600] : ? 500

Fig. 3


MAX.SCALE = 2.043573
100
200
300
400 nsec
Data File Name : ? UV.DAT
Data File Size [1(MG00] : ? 500


MAX.SCALE = 72.07626
400 nsec
Data File Name : ? UV.DAT
Data File Size [1(MG00] : ? 500


SIMULATED GRAPHIC DATA
((file = IT.DAT))
MAX.SCALE = 81.78360
100
200
300
400 nsec
Data File Name : ? UT.DAT
Data File Size [1(MG00] : ? 500

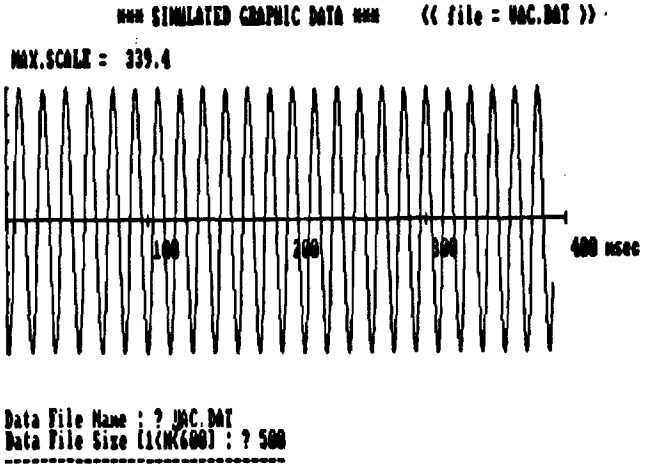
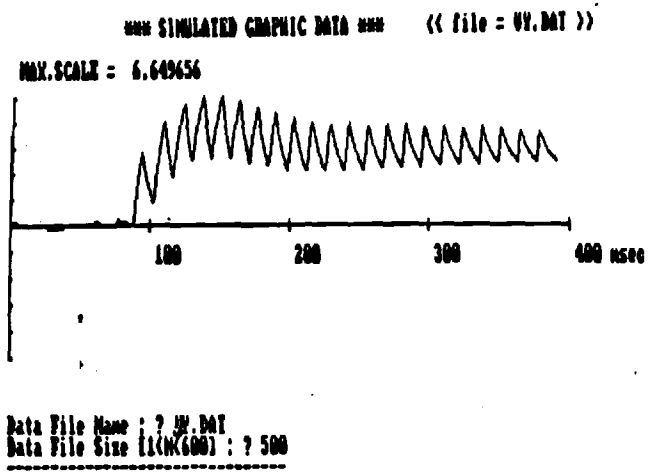
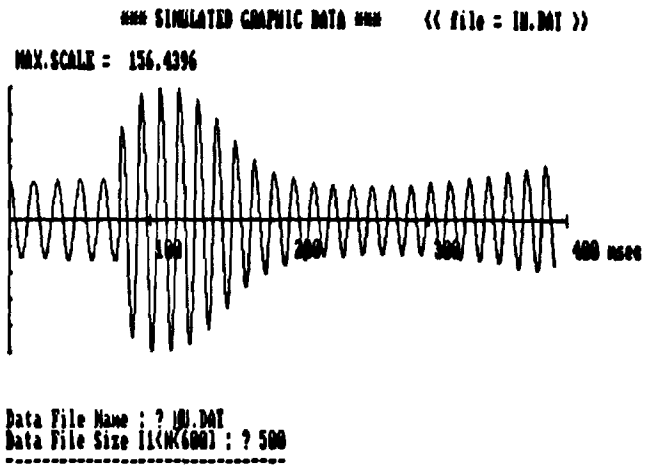
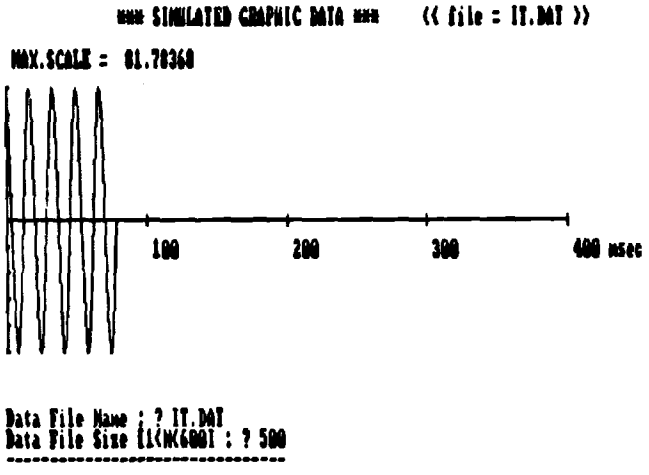

SIMULATED GRAPHIC DATA
((file = UAC.DAT))
MAX.SCALE = 339.4
400 nsec
Data File Name : ? UAC.DAT
Data File Size [1(MG00] : ? 500

Fig. 4


*** SIMULATED GRAPHIC DATA ***
<< file = VV.DAT >>
MAX.SCALE = 6.649656
100
200
300
400 msec
Data File Name : ? VV.DAT
Data File Size [1(MK600)] : ? 500


*** SIMULATED GRAPHIC DATA ***
<< file = IN.DAT >>
MAX.SCALE = 156.4396
100
200
300
400 msec
Data File Name : ? IN.DAT
Data File Size [1(MK600)] : ? 500


*** SIMULATED GRAPHIC DATA ***
<< file = IT.DAT >>
MAX.SCALE = 81.78360
100
200
300
400 msec
Data File Name : ? IT.DAT
Data File Size [1(MK600)] : ? 500

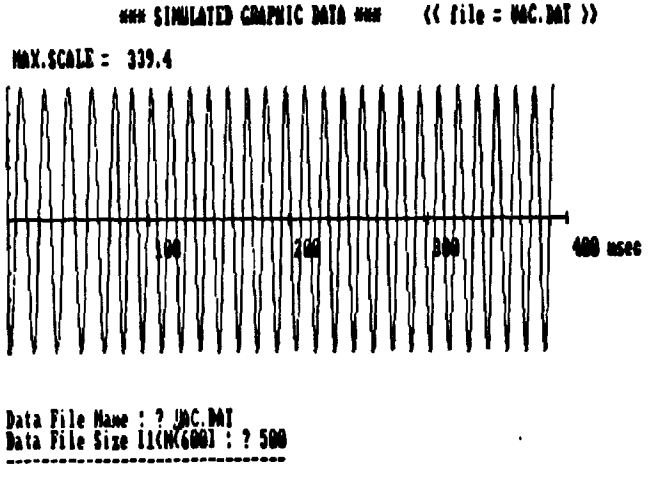

*** SIMULATED GRAPHIC DATA ***
<< file = VAC.DAT >>
MAX.SCALE = 339.4
100
200
300
400 msec
Data File Name : ? VAC.DAT
Data File Size [1(MK600)] : ? 500

Fig. 5

```
000 '*****************************************************************
010 '
020 '      THIS PROGRAM SIMULATES THE SPC INVERTER
030 '      ESPECIALLY OF THE PLL SYSTEM (Io & Vy).          G.I.T.
040 '
050 '      1986.10.11       Dr. Vachtsevanos     (EE Professor       )
060 '                       GRA   Hoon   Kang    (EE Ph.D. Student   )
070 '
080 '*****************************************************************
090 OPEN "O",#1,"VAC.DAT":OPEN "O",#2,"IU.DAT"
100 OPEN "O",#3,"IT.DAT":OPEN "O",#4,"VY.DAT"
110 OPEN "O",#5,"IO.DAT"
120 '
130 '     K1=COMPUTING INTERVAL, K2=SAMPLING INTERVAL, K3=TOTAL TIME
140 '
150 K1=1/15360:K2=12/15360:K3=9000/15360
160 READ NN : DATA 2 :REM SYSTEM ORDER FOR LINE FILTER AND LOOP FILTER
170 READ NB : DATA 1 :REM SYSTEM ORDER FOR THE BUCK STAGE SYSTEM
180 READ NU : DATA 2 :REM SYSTEM ORDER FOR THE UNFOLDER SYSTEM
190 PI=3.141592 : FLAG=0
200 DIM B(4),V(2),E(2),X(2),Y(2),V1(2),E1(2),X1(2),Y1(2)
210 DIM V2(2),E2(2),X2(2),Y2(2),V3(3),E3(3),X3(3),Y3(3)
220 DIM V4(2),E4(2),X4(2),Y4(2),V5(2),E5(2),X5(2),Y5(2)
230 DIM V6(2),E6(2),X6(2),Y6(2)
240 X(1)=-100:X(2)=0:X1(1)=0:X1(2)=0:X2(1)=0
250 X3(1)=0:X4(1)=0:X4(2)=0
260 BSIZE=1000:R=0:V=0:VY=0:CNT=0
270 VC1=0:RT=0:QT=0:DPNT=0:NPNT=0+256
280 DIM BUF(1000),LNO(2),LPO(2)
290 FOR I=0 TO BSIZE:BUF(I)=0:NEXT I
300 REM THE DIMENSION OF V,E,X,Y IS THE ORDER OF ODE
310 READ A0,A1,C1: DATA 48.35,14.67,377
320 READ A,B,C: DATA 4.6925E6,4.315,137.74
330 READ UFA,UFB,UFC,UFD: DATA 37.392,6.744E8,1.3488E5,4.5376E6
340 READ B(1),B(2),B(3),B(4):DATA 1,2,2,1
350 IMAG=14400:L=0:SHIFT=0
360 '-----------------------------------------------------------------
370 '      PROGRAM START HERE              (INITIALIZATION)
380 '-----------------------------------------------------------------
390 U=0:T=0
400 '-----------------------------------------------------------------
410 '      line filter program             (PLL)   input: VAC  output: LNO
420 '-----------------------------------------------------------------
430 B1=.5:FOR P=1 TO 4
440 REM AUXILIARY VARIABLES
450 Y1=X(2)+C1*VAC:Y2=-A1*X(2)-A0*X(1)-C1*A1*VAC
460 IF U=0 THEN 1600
470 REM INTEGRATOR INPUTS V(1),V(2),...
480 V(1)=Y1:V(2)=Y2
490 ON P GOTO 1500,1520,1520,1540
500 FOR I=1 TO NN:E(I)=K1*V(I):Y(I)=X(I):X(I)=X(I)+.5*E(I):NEXT I
510 GOTO 1560
```

```
520 FOR J=1 TO NN:X(J)=K1*V(J):E(J)=E(J)+B(P)*X(J):X(J)=Y(J)+B1*X(J):NEXT J
530 GOTO 1560
540 FOR F=1 TO NN:X(F)=E(F)+K1*V(F):X(F)=Y(F)+X(F)/6:NEXT F
550 GOTO 1580
560 IF P=2 THEN B1=1:GOTO 1580
570 T=T+.5*K1
580 NEXT P
590 T=T-K1
600 LNO(1)=X(1)
610 IF (LNO(1)>=0) AND (RT<0) THEN R=1
620 RT=LNO(1)
630 '-------------------------------------------------------------------
640 '    delay and positive edge detector        (PLL)   output: V
650 '-------------------------------------------------------------------
660 QVCC=BUF(DPNT)
670 IF (QVCC>=90) AND (QT<90) THEN V=1
680 QT=QVCC
690 '-------------------------------------------------------------------
700 '    phase comparator program                 (PLL)  output: IO
710 '-------------------------------------------------------------------
720 IF R=1 THEN IO=IO+.000227:R=0
730 IF IO>.000227 THEN IO=.000227
740 IF V=1 THEN IO=IO-.000227:V=0
750 IF IO<-.000227 THEN IO=-.000227
760 '-------------------------------------------------------------------
770 '     loop filter program           (PLL)   input: IO   output: VY
780 '-------------------------------------------------------------------
790 B1=.5:FOR P=1 TO 4
800 Y1=X1(2)+A*IO:Y2=-C*X1(2)+A*(B-C)*IO
810 IF U=0 THEN 1950
820 REM INTEGRATOR INPUTS V1(1),V1(2),...
830 V1(1)=Y1:V1(2)=Y2
840 ON P GOTO 1850,1870,1870,1890
850 FOR I=1 TO NN:E1(I)=K1*V1(I):Y1(I)=X1(I):X1(I)=X1(I)+.5*E1(I):NEXT I
860 GOTO 1910
870 FOR J=1 TO NN:X1(J)=K1*V1(J):E1(J)=E1(J)+B(P)*X1(J):X1(J)=Y1(J)+B1*X1(J):NE
  J
880 GOTO 1910
890 FOR F=1 TO NN:X1(F)=E1(F)+K1*V1(F):X1(F)=Y1(F)+X1(F)/6:NEXT F
900 GOTO 1930
910 IF P=2 THEN B1=1:GOTO 1930
920 T=T+.5*K1
930 NEXT P
940 T=T-K1
950 VY=X1(1)
960 '-------------------------------------------------------------------
970 '    8-bit counter & VCC program    (PLL)   input: VY   output: CNT
980 '-------------------------------------------------------------------
990 CNT=CNT+1+VY*920/15360
000 IF CNT>=256 THEN CNT=0
010 '-------------------------------------------------------------------
020 '    delay 1 cycle program          (PLL)   input: CNT  output: BUF
030 '-------------------------------------------------------------------
040 QVCC=CNT/256*360:BUF(NPNT)=QVCC
050 DPNT=(DPNT+1) MOD BSIZE:NPNT=(NPNT+1) MOD BSIZE
```

```
170 '     BUCK   STAGE            input: IREF, VC1        ouput: IB, VB
180 '-------------------------------------------------------------------
190 IREF=IMAG*(SIN(QVCC*PI/180))
200 B1=.5:FOR P=1 TO 4
210 Y1=-10433.33*X2(1)+1666.667*(390*IREF-VC1)
220 IF U=0 THEN 2260
230 REM INTEGRATOR INPUTS V2(1),V2(2),...
240 V2(1)=Y1
250 ON P GOTO 2160,2180,2180,2200
260 FOR I=1 TO NB:E2(I)=K1*V2(I):Y2(I)=X2(I):X2(I)=X2(I)+.5*E2(I):NEXT I
270 GOTO 2220
280 FOR J=1 TO NB:X2(J)=K1*V2(J):E2(J)=E2(J)+B(P)*X2(J):X2(J)=Y2(J)+B1*X2(J):NE
 J
290 GOTO 2220
300 FOR F=1 TO NB:X2(F)=E2(F)+K1*V2(F):X2(F)=Y2(F)+X2(F)/6:NEXT F
310 GOTO 2240
320 IF P=2 THEN B1=1:GOTO 2240
330 T=T+.5*K1
340 NEXT P
350 T=T-K1
360 IB=X2(1)
370 '-------------------------------------------------------------------
380 '   UNFOLDER  SYSTEM       input: IB, VAC          output: VC1, IU
390 '-------------------------------------------------------------------
500 L=-1:IF VAC>=0 THEN L=1
510 B1=.5:FOR P=1 TO 4
520 Y1=X3(2)+UFC*IB:Y2=-UFA*X3(2)-UFB*X3(1)-UFD*IB+UFB*VAC
530 IF U=0 THEN 2470
540 REM INTEGRATOR INPUTS V3(1),V3(2),...
550 V3(1)=Y1:V3(2)=Y2
560 ON P GOTO 2370,2390,2390,2410
570 FOR I=1 TO NU:E3(I)=K1*V3(I):Y3(I)=X3(I):X3(I)=X3(I)+.5*E3(I):NEXT I
580 GOTO 2430
590 FOR J=1 TO NU:X3(J)=K1*V3(J):E3(J)=E3(J)+B(P)*X3(J):X3(J)=Y3(J)+B1*X3(J):NE
 J
400 GOTO 2430
410 FOR F=1 TO NU:X3(F)=E3(F)+K1*V3(F):X3(F)=Y3(F)+X3(F)/6:NEXT F
420 GOTO 2450
430 IF P=2 THEN B1=1:GOTO 2450
440 T=T+.5*K1
450 NEXT P
460 T=T-K1
470 VC10=VC1:VC1=X3(1)
480 REM NEXT STEPS FOR IU
490 B1=.5:FOR P=1 TO 4
500 Y1=-37.5*X4(1)+5000*(VC10-VAC)
510 IF U=0 THEN 2650
520 REM INTEGRATOR INPUTS V4(1),V4(2),...
530 V4(1)=Y1
540 ON P GOTO 2550,2570,2570,2590
550 E4(1)=K1*V4(1):Y4(1)=X4(1):X4(1)=X4(1)+.5*E4(1)
560 GOTO 2610
570 X4(1)=K1*V4(1):E4(1)=E4(1)+B(P)*X4(1):X4(1)=Y4(1)+B1*X4(1)
580 GOTO 2610
590 X4(1)=E4(1)+K1*V4(1):X4(1)=Y4(1)+X4(1)/6
600 GOTO 2630
610 IF P=2 THEN B1=1:GOTO 2630
620 T=T+.5*K1
630 NEXT P
640 T=T-K1
650 IU=X4(1)
```

```
660 '-------------------------------------------------------------------
670 '   SIMULATION MODEL OF THE LOAD   input: IU, VS   output: VAC
680 '-------------------------------------------------------------------
690 '   FOR OUTPUT (IT)  ****************************
700 VS=339.4*SIN(120*PI*T)
710 IF FLAG =1 THEN 2890
720 B1=.5:FOR P=1 TO 4
730 Y1=-376.96*X5(1)+5235.6*(VAC-VS)
740 IF U=0 THEN 2880
750 REM INTEGRATOR INPUTS V5(1),V5(2),...
760 V5(1)=Y1
770 ON P GOTO 2780,2800,2800,2820
780 E5(1)=K1*V5(1):Y5(1)=X5(1):X5(1)=X5(1)+.5*E5(1)
790 GOTO 2840
800 X5(1)=K1*V5(1):E5(1)=E5(1)+B(P)*X5(1):X5(1)=Y5(1)+B1*X5(1)
810 GOTO 2840
820 X5(1)=E5(1)+K1*V5(1):X5(1)=Y5(1)+X5(1)/6
830 GOTO 2860
840 IF P=2 THEN B1=1:GOTO 2860
850 T=T+.5*K1
860 NEXT P
870 T=T-K1
880 ITO=IT:IT=X5(1)
890 IF FLAG=1 THEN 2910
900 IF (U>4000) AND (IT>=0) AND (ITO<0) THEN FLAG=1
910 IF FLAG=1 THEN IT=0
920 '   FOR OUTPUT VAC   ****************************
930 IF FLAG=0 THEN VAC=VS:T=T+K1:GOTO 3170
940 IF (U>=4000) THEN SHIFT =.3*PI/180*(U-4000)/10
950 VAC=339.4*SIN(120*PI*T+SHIFT):T=T+K1:GOTO 3170
960 '
970 '----------   THE FOLLOWING ROUTINE IS UNDER TESTING  ---------------
980 '
990 UU=(IU-IT)
000 B1=.5:FOR P=1 TO 4
010 Y1=X6(2)+UU:Y2=-6.3726E+07*X6(1)-4487.8*X6(2)-3472.2*UU
020 IF U=0 THEN 3150
030 REM INTEGRATOR INPUTS V6(1),V6(2),...
040 V6(1)=Y1:V6(2)=Y2
050 ON P GOTO 3060,3080,3080,3100
060 FOR I=1 TO 2:E6(I)=K1*V6(I):Y6(I)=X6(I):X6(I)=X6(I)+.5*E6(I):NEXT I
070 GOTO 3120
080 FOR J=1 TO 2:X6(J)=K1*V6(J):E6(J)=E6(J)+B(P)*X6(J):X6(J)=Y6(J)+B1*X6(J):NE
 J
090 GOTO 3120
100 FOR F=1 TO 2:X6(F)=E6(F)+K1*V6(F):X6(F)=Y6(F)+X6(F)/6:NEXT F
110 GOTO 3140
120 IF P=2 THEN B1=1:GOTO 3140
130 T=T+.5*K1
140 NEXT P
150 VAC=X6(1)*1000000!
```

```
160 '-------------------------------------------------------------
170 '   print & output to disk routine
180 '-------------------------------------------------------------
190 GOTO 3240
200 PRINT"-------------------------------------------------------------------------"
210 PRINT "U=";U,"IB=";INT(IB*1000)/1000,"IU=";INT(IU*1000)/1000
220 PRINT "IT=";INT(IT*1000)/1000,"VS=";INT(VS*100)/100,"VAC=";VAC
230 PRINT "IO=";IO,"VY=";VY
240 IF U<3000 THEN 3300
250 IF (U MOD 12)<>0 THEN 3300
260 PRINT ".";
270 'TT=T-K1
280 PRINT #1,VAC:PRINT #2,IU
290 PRINT #3,IT:PRINT #4,VY:PRINT #5,IO
300 U=U+1
310 IF U<9000 THEN 1430
320 CLOSE
330 END
```

SIMULATION STUDIES OF ISLANDED BEHAVIOR OF GRID-CONNECTED PHOTOVOLTAIC SYSTEMS

George Vachtsevanos, Member IEEE and Hoon Kang
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

ABSTRACT

A simplified computer model is presented for predicting the run-on time of a self-commutated inverter operating in a utility interactive mode. The inverter receives DC power from the photovoltaic array and delivers AC power to a local load or the utility lines. The proposed model represents the dynamics of a phase-locked loop control circuitry which is designed to destabilize the inverter operation and shut down the power conditioning subsystem when a phase discrepancy between the line and some reference signal is detected. The sustained isolated operation (or islanding) of the PV system poses a possible safety concern to utility personnel and potential damage to utility connected equipment. The computer model is implemented on an IBM PC using PASCAL and provides results compatible with experimental evidence and more elaborate computer modeling techniques.

INTRODUCTION

Over the past decade, we have witnessed an increased interest in exploiting renewable energy sources, such as solar energy, for the production of electric power. Photovoltaic arrays offer an attractive means for the direct conversion of solar energy to electrical form. An intensive R&D activity has been directed, therefore, towards the development of efficient, reliable, and cost-effective photovoltaic cells, as well as the design of interface and balance-of-system devices which are required for the distribution, storage, and consumption of the electric power, such as converters, inverters, maximum power trackers, batteries, etc. [1-3].

At an early stage of the PV development program, it was recognized that, in addition to stand-alone applications, probably the most attractive long term utilization of PV arrays is their operation in a utility interactive mode. Although institutional and economic considerations favor such an interactive operation, many technical issues must be resolved before PVs become a viable and acceptable alternative to utility generation practices. Research problems that are currently being addressed include protection and safety issues, power quality, harmonic current generation and propagation, control, islanding of dispersed sources, etc. [4-7]. "Islanding" or isolated operation usually refers to the continued generation from grid-connected photovoltaic systems following the interruption of utility power. A special case known as a "run-on" condition involves the operation of a single PV system-load combination when the power conditioning subsystem is designed to operate only in a utility interactive mode. Such islanding or run-on conditions may pose a safety hazard to utility personnel or endanger the integrity of protective and other utility equipment. Live crew personnel, working to repair a fault occurring on the grid, may mistakenly consider the load side of the line to be inactive; in fact, islanding PV sources may be feeding power back to the utility grid through power conditioning units which are normally designed to shut down when such events occur. Also, depending upon the duration of an islanding condition, automatic reconnection of the PV system may present severe resynchronization problems with consequent detrimental effects upon the integrity of utility equipment.

Experimental evidence of islanding or running-on with commercial PV systems has been provided by a number of investigators [8].

Recognizing the potential hazards, SANDIA National Laboratories, in cooperation with several power utilities and research institutions, has embarked upon an extensive testing and analytical modeling effort designed to investigate islanding phenomena. Specific objectives include the assessment of existing PCS control strategies and recommendations for modifications necessary to improve their shut down characteristics, the quantification of tolerance windows beyond which an islanding condition may not be sustained, the study of dynamic interactions between multiple islanding PV systems, and parametric sensitivity studies with varying load, PV generation, and feeder characteristics.

Researchers at Purdue University [9] have developed a hybrid computer model to study a variety of islanding configurations. Southern Company Services and Georgia Power Company are conducting an extensive laboratory and field testing program intended to validate analytical results and provide a reliable data base. The available analytical tools are extremely detailed and highly sophisticated requiring considerable skill and computational power for their implementation. As a result, they lack portability and require skilled personnel for their use.

In planning the interconnected operation of such PV systems, can utility personnel predict more readily any possible islanding events and estimate their approximate duration given the typical range of local load and power utility conditions? If an approximate but readily available tool provides any indication of possible problem situations, then a more detailed modeling and analysis effort may be undertaken to resolve these issues unquestionably.

This work is concerned with the development of a simplified analytical model of those features of the interconnected PV system-utility grid that affect significantly its isolated operation.

THE INVERTER CONFIGURATION

The power conditioning subsystem (PCS) is central to the utility-interactive photovoltaic system

operation. Figure 1 is a block diagram description of a PV utility-interactive system. The PCS is primarily responsible for converting the DC PV power into utility compatible AC power and for synchronizing and transferring that AC power safely into the utility grid. The DC to AC inversion is typically accomplished via static line- or self-commutated inverters. A line-commutated inverter utilizes the utility voltage as a means of commutation (turn-off) of transistor or thyristor-switching devices, whereas a self-commutated type uses internally generated mechanisms to perform the commutation.

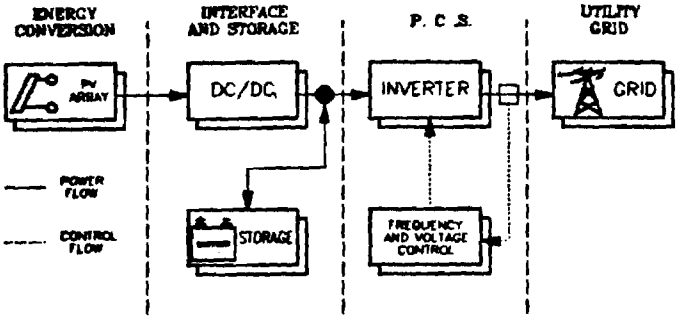


Figure 1. Block Diagram of a Typical PV-Utility Grid Interconnection

This study focuses on one such power conditioning subsystem available commercially for residential PV applications. It typifies the control philosophy adopted by most PCS manufacturers in order to provide optimum conversion while adequately handling the variable electrical conditions and safeguarding the integrity of the interconnection. A utility-interactive self-commutated, voltage-sourced inverter based on high-frequency isolation has been developed by TESLACO to provide a photovoltaic to utility interface at the residential power level of 4 kW. Figure 2 is a block diagram of the TESLACO T-4 kW-A power stage and control circuitry. A detailed description of the TESLACO inverter may be found in the excellent paper by A. Cocconi, et al. [10]. The DC power from the solar array is converted to a full wave rectified sinusoidal waveform through the push-pull buck stage converter. The buck stage also provides isolation between the DC and AC sides of the circuit through the use of a 20 kHz compact and lightweight transformer. In the second stage of the inverter, alternate half sine waves are inverted by unfolding to form the complete output sine wave that is matched to the 240 volt AC line.

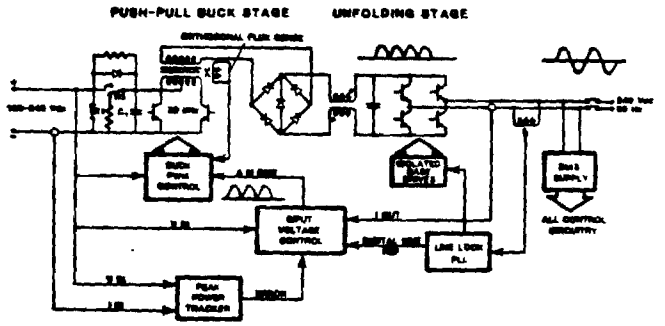


Figure 2. Block Diagram of the TESLACO T-4 kW-A Power Stage and Control Circuitry

The control circuitry performs a multiplicity of functions ranging from tracking the maximum power point of the PV array to generating the PWM base drive signals and controlling the output unwrapper transistor bridge. A brief overview of the control strategy follows, emphasizing those features that are relevant to an islanded operation of the inverter. The peak power tracker follows the array peak-power point by continually monitoring the ripple on the DC input. Its operation is based upon balancing the symmetry of the 120 Hz power ripple signal. The unfolding transistor control circuitry monitors the current in each of the four transistors and acts to turn them off rapidly in the event of a fault condition on the line. The output current feedback control circuit measures the average AC output current of the system and modulates the signal to the buck control circuit so as to maintain it at the desired level.

The most crucial control component responsible for shutting down the transistor bridge in the event of a utility disconnect is the line lock phase-locked loop (PLL). A reference rectified sine wave voltage is generated internally and locked to the line by use of the digital PLL circuitry. Current delivered to the line is automatically in phase with the voltage, once it is brought into synchronization with the line through the PLL circuitry. A phase discrepancy between the line and reference signal is used to destabilize the loop upon line disconnection resulting ultimately in inverter shutdown. Since the PLL plays a significant role in interrupting the power flow from the inverter to the utility grid under fault conditions, its operational characteristics will be described more fully in the next section.

MODEL DESCRIPTION

Figure 3 shows a block diagram of the PLL circuit. The phase comparator is preceded by an integrator in order to smooth our fast disturbances on the AC line. The 90° phase shift of the integrator is compensated by decoding the 90° address of the EPROM clock. The loop filter is the only part of the circuit that introduces any dynamics into the modulation transfer function. The loop filter uses an ordinary lead-lag network for phase compensation and is represented by its transfer function. The output of the phase comparator, I_o, is equal to +0.227 mA (-0.227 mA) during the period of time when the reference pulse, R, leads (lags) the pulse, V, and is zero otherwise. Thus, during steady state operation at 60 Hz, R and V are in phase so that I_o is zero.

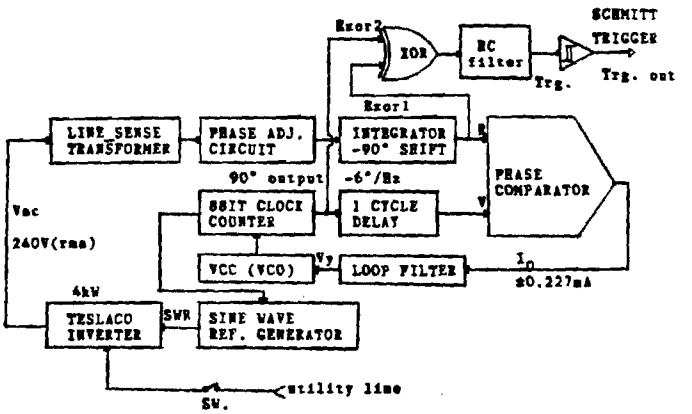


Figure 3. Block Diagram of the Line Locking PLL

The output of the loop filter is described by the following relation:

$$\frac{V_y(s)}{I_o(s)} = \frac{1.47 \times 10^5}{S} + \frac{33000}{1 + S(7.26 \times 10^{-3})}$$

If the loop filter output, V_y, is zero, then the voltage control oscillator produces a pulse train of 15.36 kHz (60 Hz x 256). When V_y is positive (negative),

then the VCO produces a frequency deviation of Δf = (920 Hz) x (V_y in volts) with a corresponding increase (decrease) in the pulse train frequency. Thus, the comparator-filter combination acts as follows: If the reference signal R leads V, the output of the phase comparator becomes positive causing V_y to increase. This increases the VCO frequency which, in turn, causes the pulse V to occur earlier as desired. The opposite situation occurs if R lags V.

The 8-bit clock produces two output signals: the first one is the sine wave reference (stored in ROM) which, for all practical purposes, may be represented as a continuous signal by the expression

$$SWR(t) = 1|\sin[\theta_{VCO}(t)]|$$

where

$$\theta_{VCO}(t) = \frac{\text{count}}{256} \times 360^\circ$$

is the angular equivalent of the counter output whose output increments by 1 at each input clock pulse.

A second (÷ 256) output is provided which is equal to a logic "1" whenever $90^\circ < \theta_{VCO}(t) < 270^\circ$. This second output is fed into a 1.00 cyle time delay. Although this one cycle time delay produces no locking angle error at exactly 60 Hz, it does introduce a transport lag in the control loop. It is exactly this 1.00 cycle transport delay that introduces a right half-plane pole of the loop gain function ensuring that the PLL will be unstable when the AC line is disconnected. As a matter of fact, the loop gain transfer function has a left half-plane pole without the 1.00 cycle time delay resulting in a stable system. The delay is, therefore, intentionally designed into the system in order to bring about the unstable condition whenever R and V are out of phase. Finally, under free run conditions, the line sense transformer is essentially connected directly to the digital sine generator via the power stage of the inverter and a phase-adjust circuit is introduced to balance out phase contributions between the internal reference wave and the terminal voltage.

The PLL circuitry of the inverter detects when the line deviates from its nominal frequency or voltage by more than a preset value and then initiates the orderly inverter shutdown. In the case of an isolated or islanded condition, the inverter must be capable of detecting the absence of the line voltage in order to initiate the shutdown procedure. Consider the typical situation of a nonunity power factor load depicted in Fig. 4. Assume that the reactive load is partly supplied from the utility. Upon the line disconnection, the output voltage will be shifted in phase from the inverter's internal equivalent voltage source, thereby producing the phase shift shown in the figure. This phase discrepancy is detected by the PLL as a shift in frequency. If the shift in frequency is already beyond the preset limit (1.0 Hz or 6° phase shift), the inverter is forced to shut down. When the

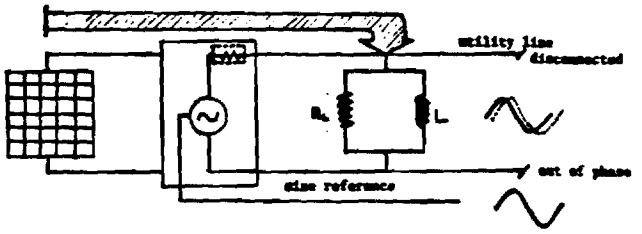
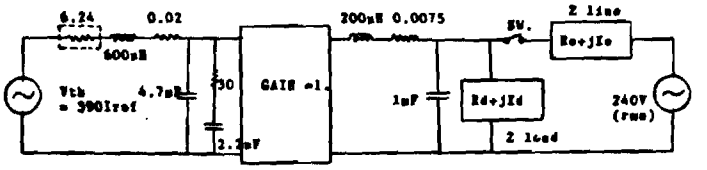


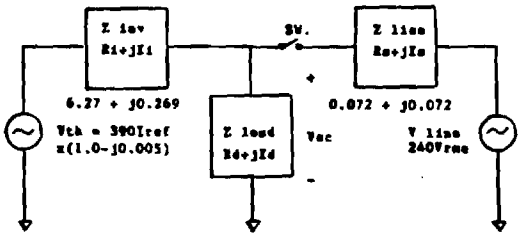
Figure 4. Effect of Utility Line Disconnection on Phase Shift

phase shift is initially insufficient to trigger the inverter shutdown, then the loop instability, as described above, assists in building up the phase difference until the critical condition is reached again. The time duration between the utility disconnect and the inverter shutdown is defined as the run-on time.

For a simplified representation of the PV system-utility grid under islanded conditions, attention must be focused, therefore, upon the dynamic behavior of the PLL circuitry responsible for the regenerative phase difference instability, on one hand, and the determination of the initial phase difference between the internal reference waveform and the terminal voltage upon disconnection of the utility line. Other major components of the power conditioning subsystem (power tracker, unfolder, etc.) do not seem to influence significantly the destabilizing behavior of the inverter. The initial phase difference is estimated approximately from a steady state representation of the buck stage-unfolder-load-utility line combination. Refer to Fig. 5(a). The first part of the circuit is a Thevenin equivalent representation of the buck power stage with voltage feedforward coupled with current feedback. The purpose of the current feedback is to provide the inverter with a nondissipative resistive output impedance of approximately 6.24 Ω. In addition to correcting for distortion in the AC line waveform, this output impedance is used to control the magnitude of the AC output current. For simplicity, the unfolder circuit is represented as a unity gain device. The remaining components in Fig. 5(a) represent filtering, the local loading, and a Thevenin equivalent of the utility line at the point of connection. The reference current, I_{ref}, may be varied by varying the input solar insolation to the PV array. Figure 5(b) shows the equivalent steady state representation of 5(a) after some simple manipulations.



(a)



(b)

Figure 5 (a) and (b). Steady State Equivalent Circuit of Block Stage-Unfolder-Load-Utility Line

Figure 6 is a flow chart of the computer program. The user is requested to input the following program data:

- The load impedance, $R_d + jX_d$
- The line impedance, $R_e + jX_e$
- The level of solar insolation, P_I, in mW/cm^2.

The line voltage is taken to be 240 v.rms. The run-on time may be computed for various values of load impedance and solar insolation in order to obtain a reasonable estimate of major trends which may establish those critical conditions in the PV-utility grid system which maximize the run-on time.

SIMULATION RESULTS AND CONCLUSIONS

The simplified model was tested with a variety of input data. Of special interest was the response of the model to load conditions which produce the longest run-on time. The validity of the simplifiying assumptions was checked by comparing results with those obtained with more detailed dynamic models and available experimental data under similar loading conditions. Finally, the sensitivity of the model to varying input conditions was verified by a series of simulation runs with load impedances and input solar insolation levels ranging over reasonable bounds.

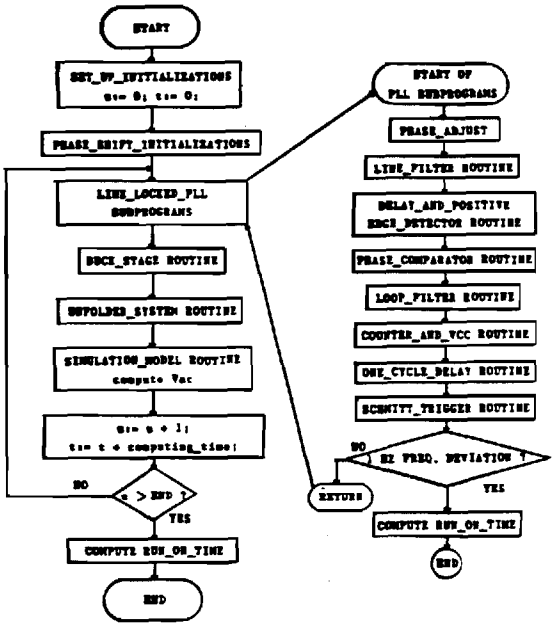


Figure 6. Flow Chart of Computer Program.

Figure 7 shows a computer printout for a typical model run. The load impedance is taken to be equal to 16.050 +j5.590 ohms which corresponds to a load power of 3195.25 watts and 1112.86 vars. The first waveform shows initially the variation of the line current I_t for a period of approximately two and one-half cycles to the point where the line is disconnected. From then on the waveform represents the loop filter output, V_y. The phase difference between the reference signal and the 8 bit clock counter is fed to an exclusive OR whose filtered output is shown as the third waveform in Fig. 7. The final two rectangular waveforms are the actual reference and clock output waveforms. The run-on time is estimated to be in this case approximately 495 msec (31 cycles). This agrees with results reported from hybrid or detailed digital simulations under similar conditions. Further simulation results indicate that as the reactive component of the load increases, the corresponding run-on time decreases. Maximum run-on time is obtained when the load matches fairly closely the generation.

Some further fine tuning of the dynamic PLL components is required in order for the model to reproduce reasonably experimental test results.

A similar modeling approach is being pursued for the representation of other commercial inverter designs. It is hoped that this simplified modeling approach of a complex system, like the DC to AC inverter with its associated utility interface, will provide utility personnel with a viable tool for assessing the potential safety and production hazards arising from the interconnected operation of PV systems.

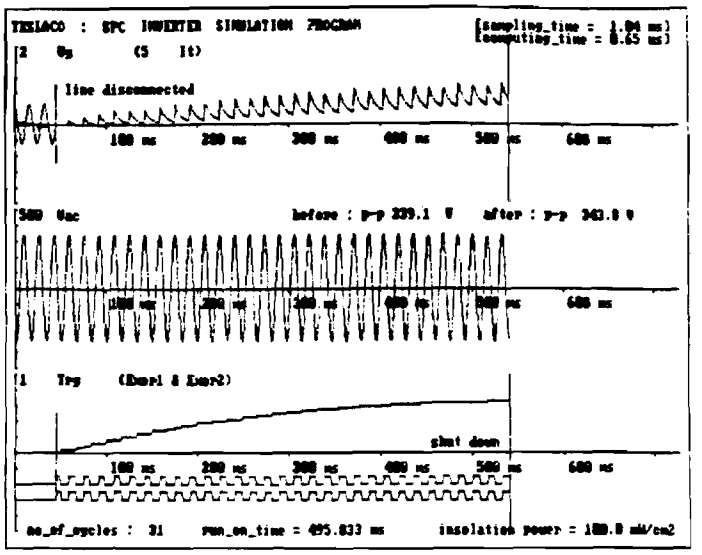


Figure 7. Typical Waveforms Obtained with the Proposed Simplified Model

ACKNOWLEDGEMENT

We wish to express our sincere appreciation to Drs. Wasynczuk and Krause of Purdue University for their helpful discussions and for making available to us the results of their hybrid simulations. Our thanks also to Georgia Power Company and Southern Company Services for their support in the contact of this work.

REFERENCES

1. S. Krauthamer, et al., "Photovoltaic Power Conditioning Subsystem: State of the Art and Development Opportunities," DOE/ET-20356-9, Jan. 1984.
2. T. Key, "Power Conditioning Technology for Grid-Connected Applications," in Fourth Photovoltaic Systems Definitions and Applications Projects Integration Meeting, Albuquerque, NM, April 1983.
3. J.W. Walton, "Advanced Converter Technology," DOE Final Report No. FCR-6846, 05/23/79-12/31/84.
4. G.J. Vachtsevanos, "Dispersed Generation - An Overview of Issues and Approaches," Proc. of MECO '83-EES '83 International Symposium, August 1983.
5. H. Chestnut, et al., "Monitoring and Control Requirements for Dispersed Storage and Generation," IEEE Trans. on Power Apparatus and Sytems, Vol. PAS-101, No. 7, July 1982.
6. Systems Control, Inc. and J.B. Patton, "Interconnecting DC Energy Systems: Responses to Technical Issues," EPRI AP/EM-3124, June 1983.
7. M. Kuliasha and T. Reddoch, "Research Needs for the Effective Integration of New Technologies into the Electric Utility," Proc. of Conference sponsored by DOE, CONF-820772, May 1983.
8. G.H. Atmaram, et al., "Test Results of Islanding Experiments on Grid-Interactive Residential Power Conditioners," Proc. of 18th IEEE Photovoltaic Specialists Conference, Las Vegas, NV, Oct. 22-25, 1985.
9. O. Wasynczuk, et al., "Dynamic Simulation of Dispersed, Grid-Connected Photovoltaic Power Systems: System Studies," SAND 83-7019 Report, March 1985.
10. A. Cocconi, et al., "High-Frequency Isolated 4 kW Photovoltaic Inverter for Utility Interface," Power Conversion International, May 1984.

FINAL REPORT

PROJECT NO. E-21-628

INVESTIGATION OF POTENTIAL ISLANDING OF DISPERSED PV SYSTEMS

(DEVELOPMENT OF SIMPLIFIED ANALYTICAL MODELS FOR POWER CONDITIONING SYSTEMS)

By

George Vachtsevanos
Hoon Kang

Submitted to

Georgia Power Company
Solar Operations
7 Solar Circle
Shenandoah, GA 30265

June 1987

Georgia Institute of Technology
A Unit of the University System of Georgia
School of Electrical Engineering
Atlanta, Georgia 30332-0250

TABLE OF CONTENTS

INTRODUCTION

This final report details the development of simplified computer models for the American Power Conversion Corporation's (APCC) Sunsine 2000 photovoltaic power conditioning unit (PCU) and the TESLACO - Optimum Power Conversion Model T - 4 kW - A photovoltaic power conditioning subsystem. Simulation studies are described using the simplified models and results are compared to experimental data as well as information available from detailed computer models.

The primary objective of this effort is to produce simple models that represent the dynamic behavior of the PCU under run-on conditions. The focal point of this endeavor is, therefore, to highlight those features of the inverter circuitry which are crucial in effecting inverter shut-down whenever the utility line is disconnected from the PCU-load combination.

This work is motivated by the need to provide utility personnel with a portable and easy-to-use tool to predict any possible islanding events and estimate their approximate duration given the typical range of local load and power utility conditions. If such an approximate but readily available tool provides an indication of possible problem situations, then a more elaborate modeling and analysis effort may be undertaken to resolve these issues unquestionably.

In most situations, the modeling results reported herein an conservative providing the user with significant trends as load conditions vary. Whereas more sophisticated computer models must exhibit a predictive capability compatible with actual laboratory or field results if they are to be used as tools in assessing design and safety concerns, a simplified approach can only claim results with a reasonable resemblance to the actual case.

Slow inverter dynamics or nonessential components to the run-on condition are neglected. The inverter designer, anticipating the need to shut the device down under emergency conditions, invariably incorporates electronic circuits which sense the resulting operating discrepancy and activate appropriate shut-off sequences. A simplified analytical model must, therefore, focus upon those features of the interconnected PV system-utility grid that affect significantly its isolated operation.

Part I of this report describes the APCC inverter simplified model and associated simulation results while Part II is devoted to a brief description of the TESLACO inverter modeling effort.

PART I

A SIMPLIFIED ANALYTICAL MODEL FOR THE APCC PHOTOVOLTAIC POWER CONDITIONING SYSTEM

I.1 General Description

Detailed electrical diagrams of commercial power conversion systems are usually considered proprietary and are not available in the open literature. Recently, Purdue University developed a hybrid computer model of the APCC Sunsine 2000 photovoltaic power conditioning unit through the assistance and cooperation of the APCC technical staff. The Purdue report [1], detailing this modeling effort, formed the only available source of information regarding the Sunsine 2000 for the development of a simplified analytical tool. Mathematical descriptions of key inverter subsystems (the line filter, the loop filter, etc.) were taken directly from the Purdue development since actual circuit diagrams are still considered proprietary. The results of this work are, therefore, dependent upon the credibility of the Purdue model. In general, for most practical situations, the computer results seem to match fairly well experimental data obtained from laboratory and field studies conducted by Georgia Power Company.

Figure I-1, reprinted from Reference [1], is a simplified electrical diagram of the APCC Sunsine PV system. The DC system consisting of the PV array and a filter capacitor is connected to a down converter which produces an output current closely resembling a full wave rectified sine wave. The unwrapper circuit produces a sinusoidal output at the utility side of the isolation transformer by inverting every other half cycle of the down converter output.

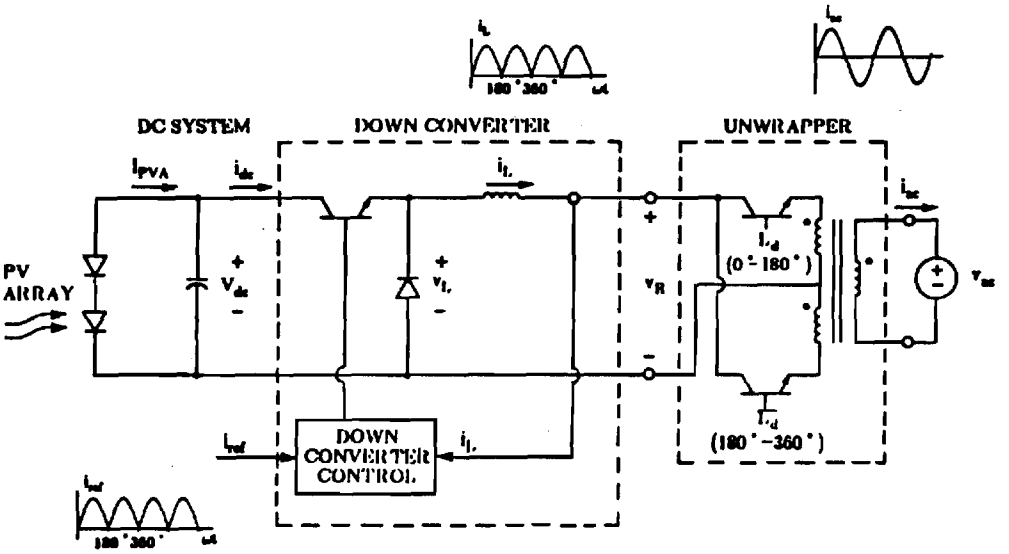


FIGURE I-1. SIMPLIFIED ELECTRICAL DIAGRAM OF APCC INVERTER.

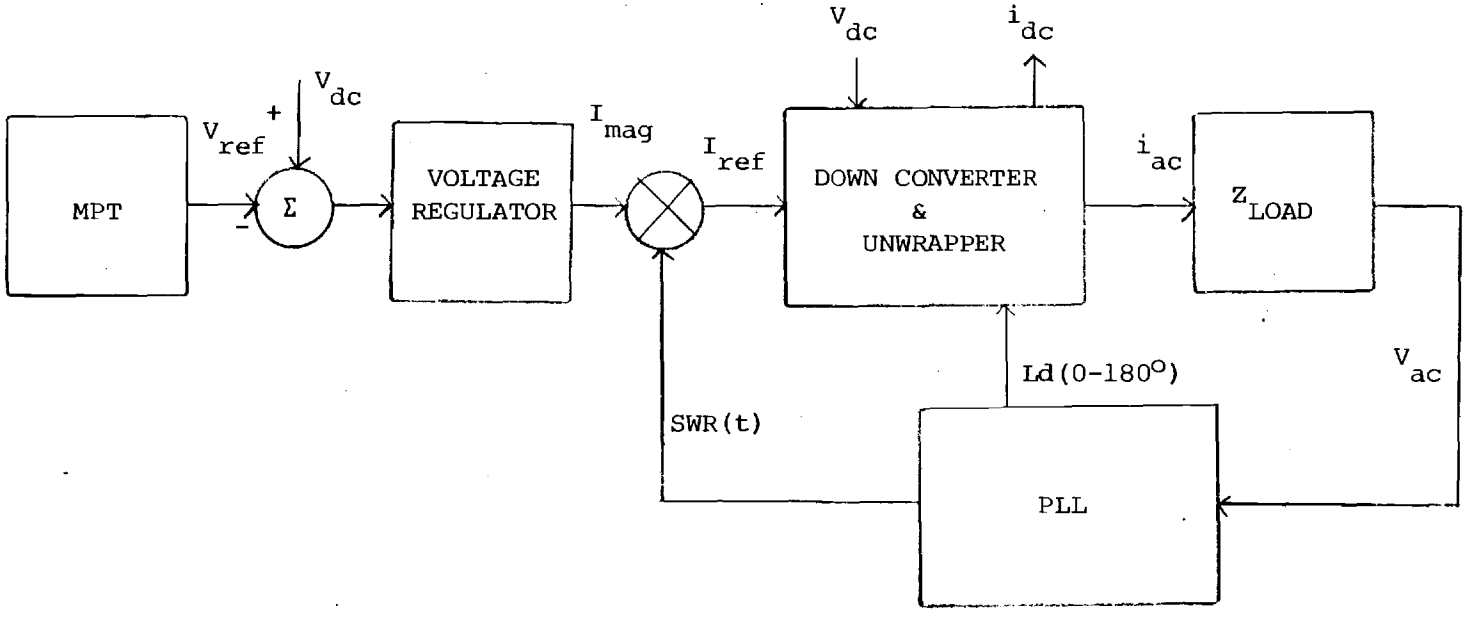


FIGURE I-2. SIMPLIFIED BLOCK DIAGRAM OF THE CONTROL SYSTEM.

The inverter AC output current is phase locked to the measured line voltage. A simplified block diagram of the inverter control system is shown in Figure I-2. A Maximum Power Tracker (MPT) assures that maximum power is extracted from the array under varying environmental conditions. The voltage regulator adjusts a magnitude command signal, I_{mag}, which is tracked by the down converter during normal operation and is a measure of the peak AC current injected into the utility.

The function of the PLL is to provide a rectified sinusoidal reference signal and to control the switching of the unwrapper. A more detailed functional block diagram of the phase locked loop control system is shown as Figure I-3. For normal operation, the multiplying digital to analog converter (MDAC) produces a rectified sinusoidal reference signal which must be synchronized with the line voltage. To accomplish this, the frequency of the voltage controlled clock (VCC) is varied in proportion to a control signal V_ω. When $V_\omega = 0$, the output frequency of the clock is approximately 30.72 kHz. At each clock pulse, the counter is incremented with a maximum count of 512. Thus, the frequency associated with a complete counter cycle is 60 Hz. As the control voltage V_ω varies from -10 to +10 volts, the frequency of a complete counter cycle will vary between 59.073 and 60.927 Hz. Synchronization is achieved using a feedback circuit which increases or decreases the control voltage whenever the counter output leads or lags the line voltage. Details of the PLL VCC and 9 bit counter are depicted in Figure I-4.

Synchronization of the counter to the line frequency is achieved by monitoring the phase difference between the counter output and the line voltage. The filtered line voltage is fed to an A/D comparator whose output represents one of two inputs to an exclusive OR (XOR) phase detector. Thus,

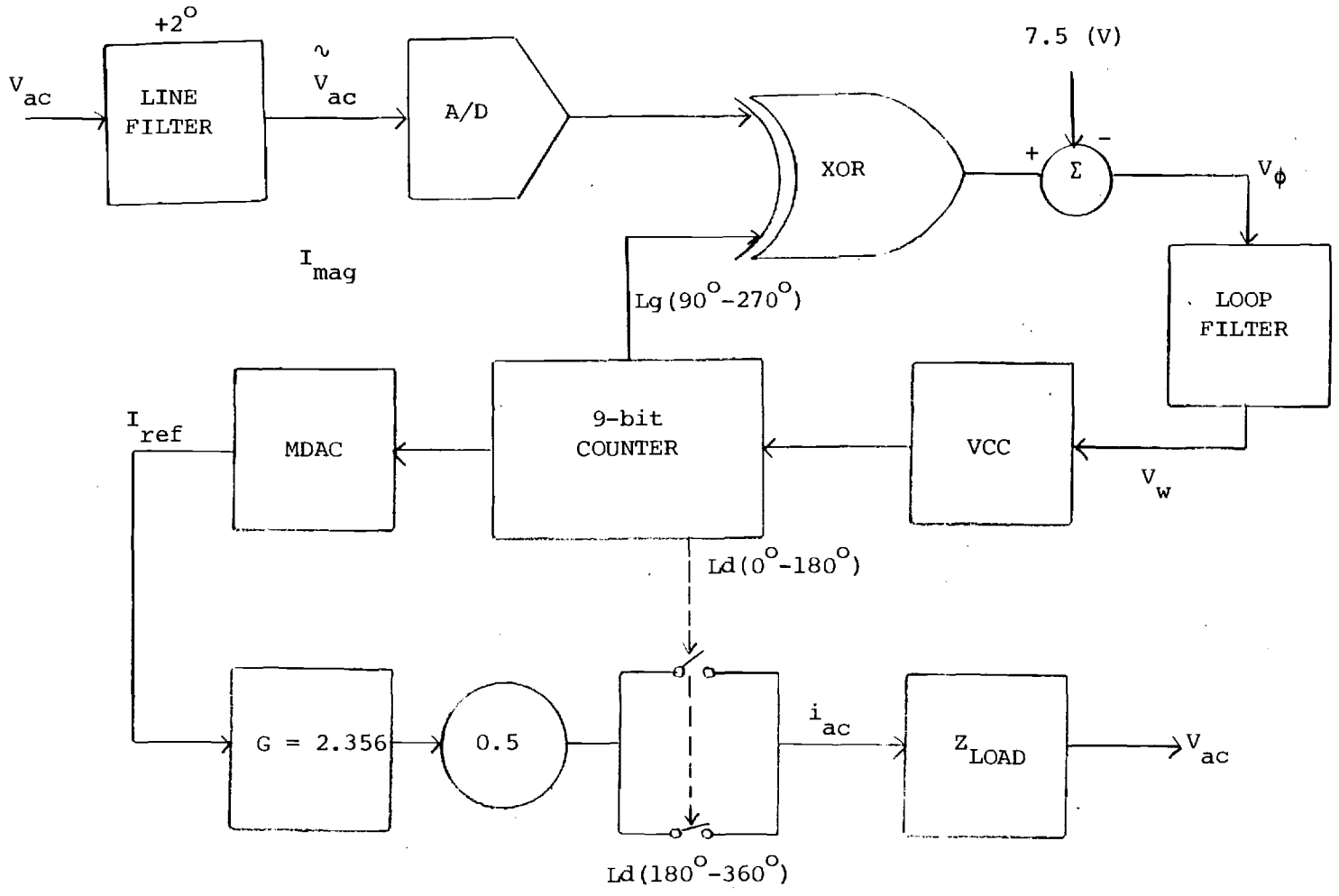


FIGURE I-3. BLOCK DIAGRAM OF THE APCC PLL CIRCUITRY.

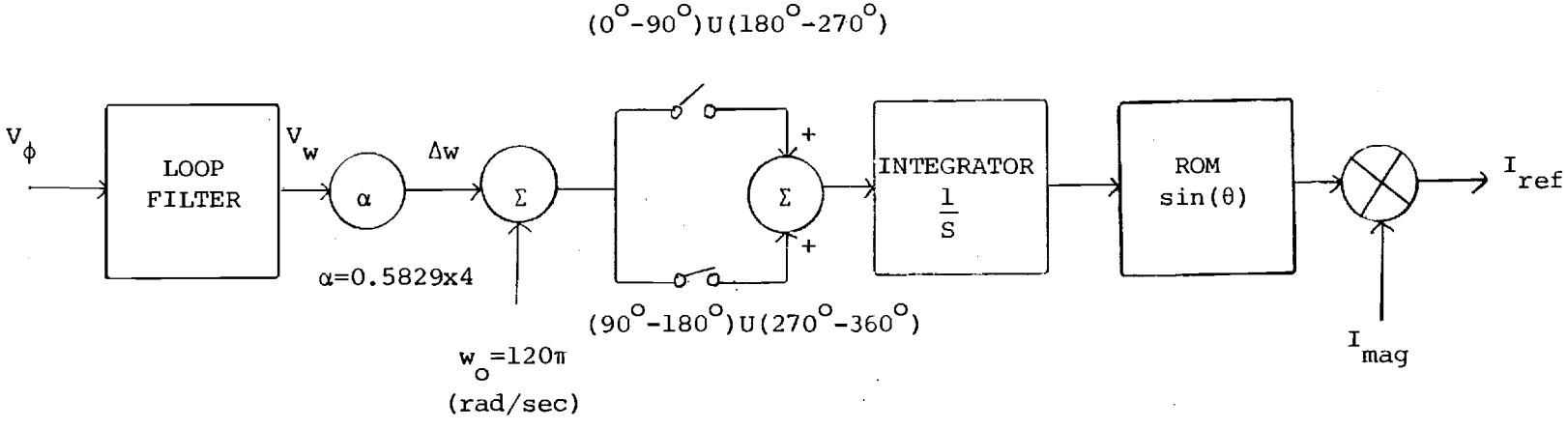


FIGURE I-4. BLOCK DIAGRAM OF THE PLL VCC AND 9-BIT COUNTER

the average value of the XOR output is a measure of the phase difference between the angular state of the counter and the line voltage.

I.2 Islanded Operation

Let us consider the interconnected operation of the PV system with the utility grid as shown in Figure I-5. Of interest is to observe the operation of the PV system in the event that the circuit breaker is suddenly opened. Particularly, it is important for operational and safety considerations to determine conditions under which the unit will run-on for the maximum possible time. As a matter of fact, if the load current is carefully adjusted so that all of the current supplied by the inverter is consumed by the load, then the opening of the breaker will not be sensed by the inverter protective control circuits. In this case, the inverter will, theoretically at least, run-on indefinitely. Under all other load conditions, the PLL and its associated control circuitry undertakes the task of sensing the appropriate phase angles before and after line disconnection and tripping a phase error detector whenever the phase error between the angular state of the phase locked oscillator and the filtered line voltage reference becomes greater than 2°. Thus, the dynamics of the PLL circuit are primarily responsible for the operation of the inverter under islanded conditions. For that reason, attention is focused below in the modeling effort on the behavior of the PLL circuitry.

Let us examine the action of the control circuit at the time of line disconnection. Let

$|\phi_e|$ = absolute phase error in degrees

$|\phi_i|$ = absolute phase difference between inverter voltage and current.

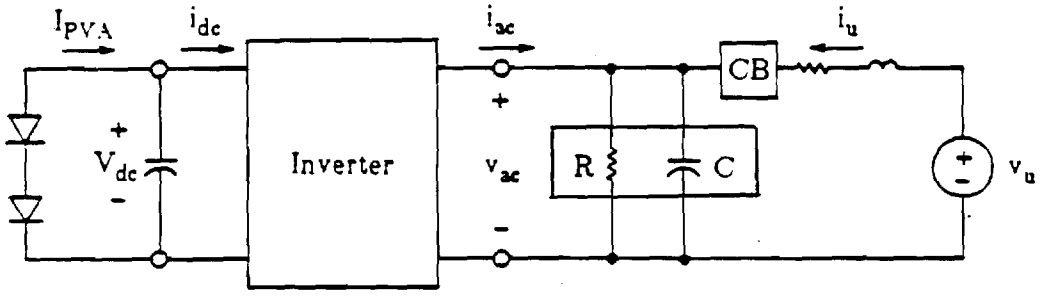


FIGURE I-5. PV-INVERTER-UTILITY INTERCONNECTION.

During steady state conditions, before the circuit breaker is opened, $|\phi_i|$ is the phase lead angle introduced by the line filter which is very close to 2°. The phase error $|\phi_e|$ is, of course, zero during steady state conditions. When the circuit breaker is opened, the value of $|\phi_i|$ is determined by the load angle, ϕ_L, of the passive load to which the inverter connects. Upon opening of the circuit breaker, $|\phi_e|$ will jump to the value such that $|\phi_L + \phi_e| = 2°$. Thereafter, the response of $|\phi_i|$, $|\phi_e|$, and V_ω is primarily determined by the characteristics of the line filter, the load and loop filter associated with the phase locked loop.

The most significant design feature of the inverter involves a resonant peak of the line filter at 60 Hz. Thus, the phase load of the line filter increases as the frequency increases and at a significantly higher rate than that of the load (if it is capacitive) such that the resulting phase error increases as the frequency increases. A classic "positive feedback" instability results since the AC system frequency will increase at a higher rate as the phase error increases.

Details of the phase error detector circuit responsible for disabling the inverter are shown in Figure I-6. If the load is resistive or inductive, then the initial jump in $|\phi_e|$ will always be greater than 2° which is the trip point of the phase error detector resulting in almost instantaneous shutdown. The same condition will result if the load is capacitive but the magnitude of load angle is between 0° and 2°. The positive feedback instability is, therefore, essential for the shutdown mode only in the case of a capacitive load with a phase angle between 0° and 2°.

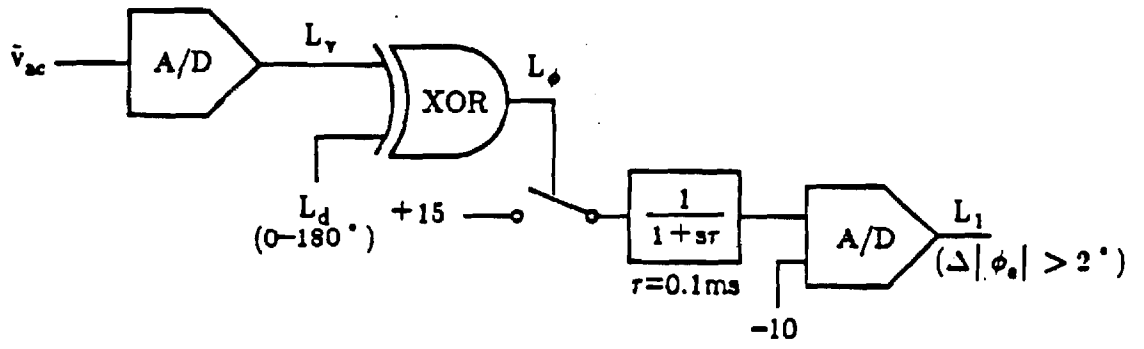


FIGURE I-6. THE PHASE ERROR DETECTOR CIRCUIT.

I.3 The Simplified Computer Model

Special attention is required in modeling the dynamics of the line filter and the loop filter of the PLL circuit. The significance of the first one in the run-on mode was demonstrated in the previous section. Modeling of the remaining control circuit components is straightforward and follows classical approaches.

I.3.1 The Line Filter Implementation

The line filter is implemented by a fifth order state equation whose transfer function is as follows:

$$H_{lf}(s) = \frac{9.803934E7*(s-2.8748E-14)(s^2+92.79882s+144998.1)}{(s+71005.7)(s+2511.29)(s+61.87084)(s^2+186.6827s+149752.5)}$$

In Figures I-7 and I-8, the magnitude and phase responses are plotted using a CC program package. As it was pointed out previously, the positive group delay at 60 Hz makes the PLL system unstable.

In state variable form, the equations are written as:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -70970.59 & -2303.92 & 1470.59 & 0 & 0 \\ -104.7237 & -104.7237 & 66.8449 & 0 & 0 \\ 1470.59 & 1470.59 & -2597.43 & 74.23905 & -4925.76 \\ 0 & 0 & 46.39941 & -46.39941 & 3078.60 \\ 0 & 0 & 46.39941 & -46.39941 & -46.39941 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

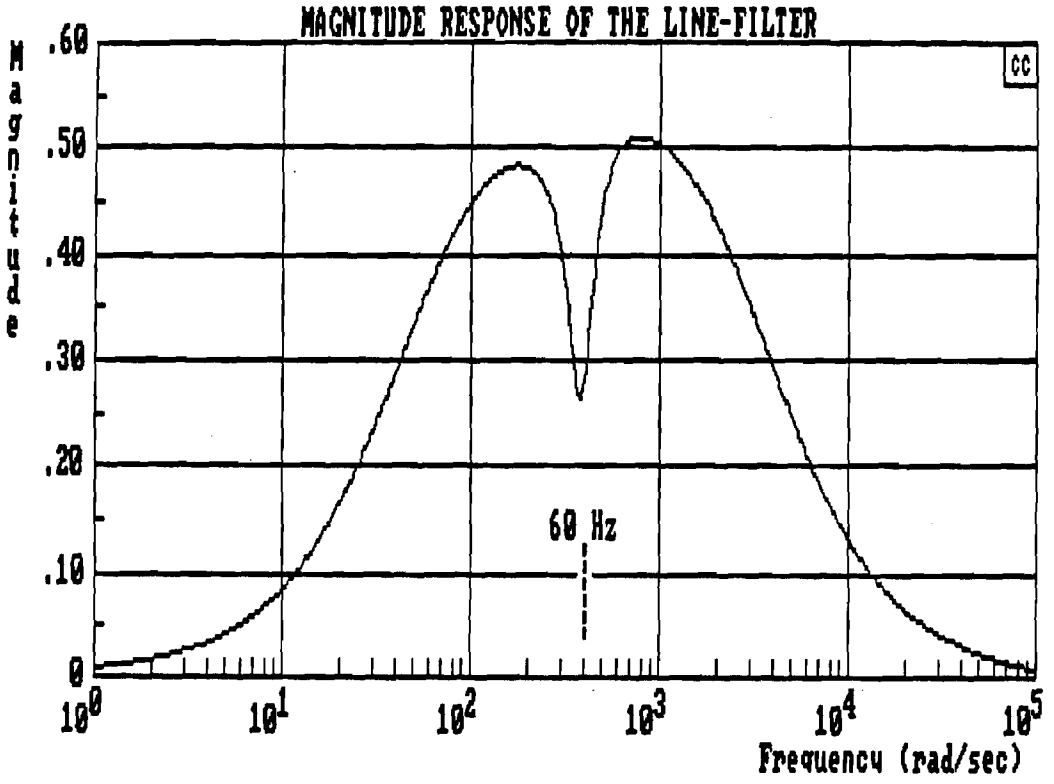


FIGURE I-7. THE MAGNITUDE RESPONSE OF THE LINE FILTER.

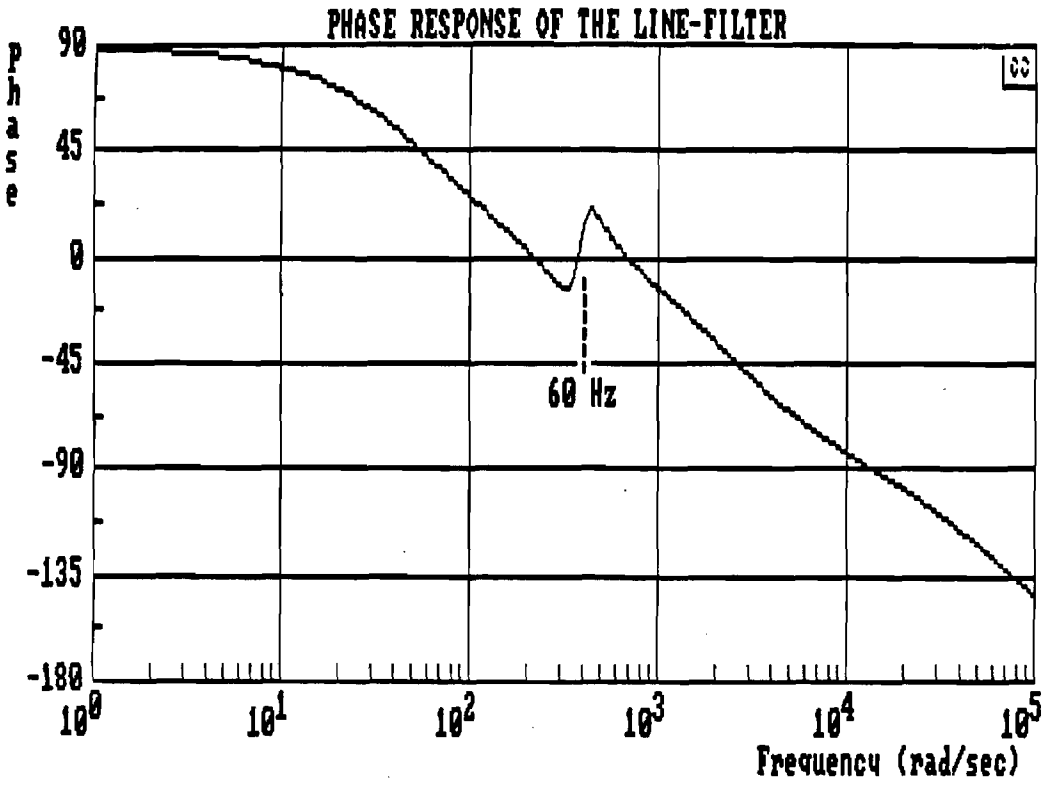


FIGURE I-8. THE PHASE RESPONSE OF THE LINE FILTER.

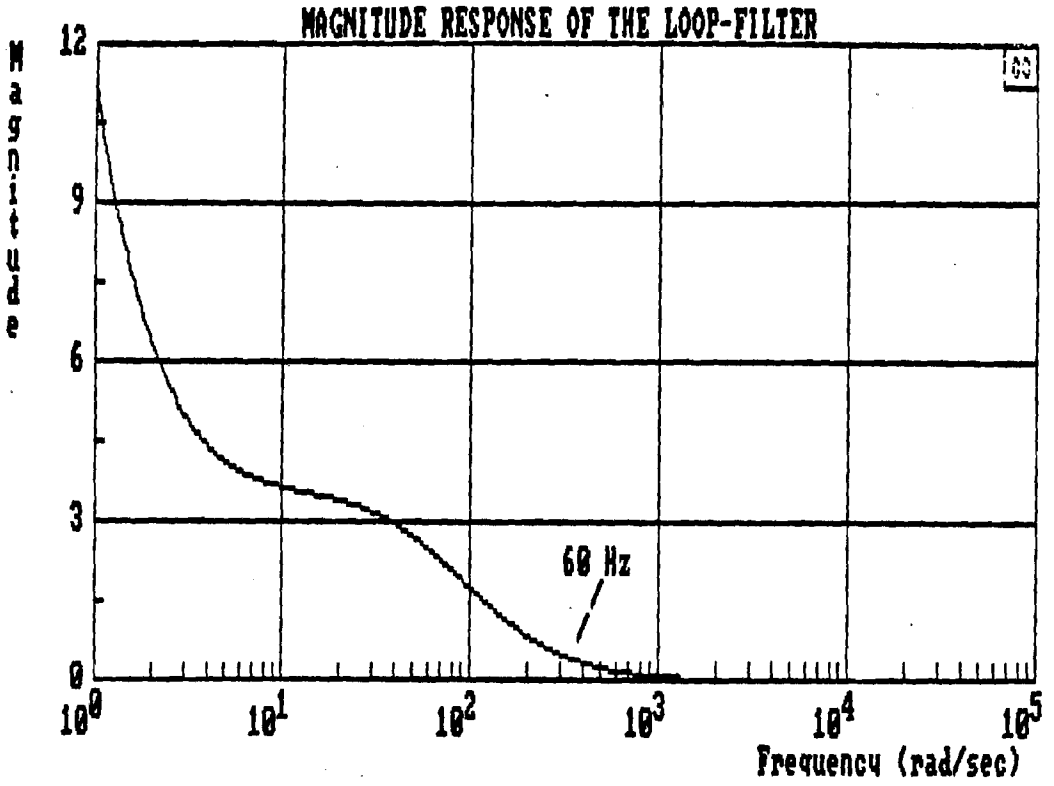


FIGURE I-9. THE MAGNITUDE RESPONSE OF THE LOOP FILTER.

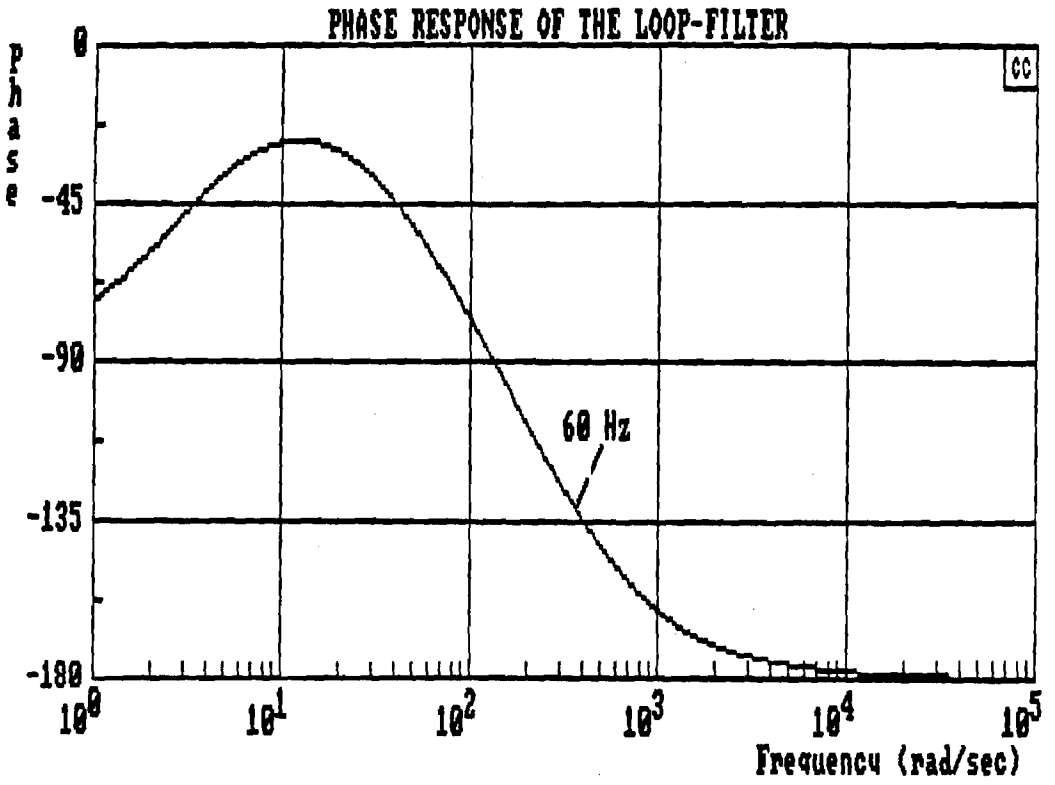


FIGURE I-10. THE PHASE RESPONSE OF THE LOOP FILTER.

$$+ \begin{bmatrix} 66666.67 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} v_{ac}$$

$$\bar{v}_{ac} = [0\ 0\ 1\ 0\ 0]*\underline{x}$$

where $\underline{x} = [x_1\ x_2\ x_3\ x_4\ x_5]^T$. The poles of the line filter transfer function are:

$$s_1 = -61.87084$$

$$s_2 = -93.34133 + j375.5528$$

$$s_3 = -93.34133 - j375.5528$$

$$s_4 = -2511.29$$

$$s_5 = -71005.7$$

I.3.2 The Loop Filter Implementation

The loop filter can be implemented by a third order state equation whose transfer function is:

$$H_{lp}(s) = \frac{67188.53*(s+3.002979)}{s(s+303.0303)(s+62.57823)}$$

In Figures I-9 and I-10, the magnitude and phase responses are plotted using the CC program package. The state representation is as follows:

$$\frac{d}{dt} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1.89631E4 & -3.656085E2 \end{matrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} + \begin{matrix} 0 \\ 6.314712E3 \\ -2.289749E6 \end{matrix} V_\phi$$

$$V_\omega = [10.64 \quad 0 \quad 0]*\underline{x}$$

where $\underline{x} = [x_1 \; x_2 \; x_3]^T$. The poles of the loop filter are:

$$s_1 = -62.57823$$
$$s_2 = -303.0303$$
$$s_3 = 0$$

V_ϕ is the input to the loop filter.

I.3.3 The Line Impedance and the Load Impedance

The line impedance is modeled as the series combination of a resistor (with a nominal value of 0.072 ohm) and an inductor (0.072 ohm). The load impedance is specified by the user.

I.4 Simulation Studies

Figure I-11 depicts a flow chart of the simulation program. The following conditions are assumed:

(1) The inverter is operating in steady state before line disconnection so that the average value of the loop filter output V_ω is zero.

(2) The circuit breaker is opened when the utility current, $I_u(t)$, goes from a negative to a positive zero crossing.

(3) The initial value of $V_\omega(t)$ is adjusted to conform to condition (1).

In order to compare model results with test data available through GPC, simulated conditions were initially set to match those of the GPC tests. The system current, $I_{AC}(t)$, was chosen to be 23.56 amperes while the inverter

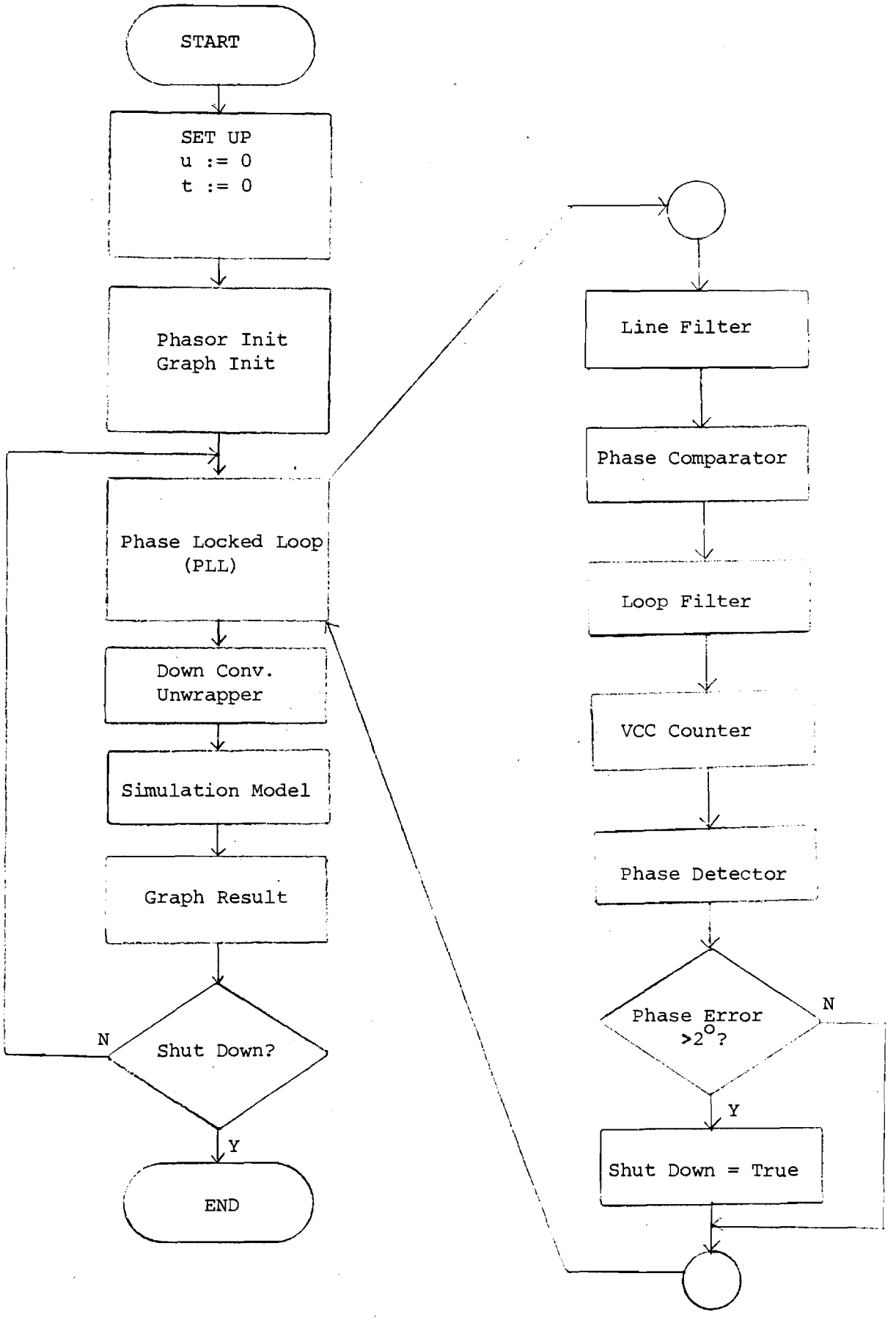


FIGURE I-11. FLOW CHART OF APCC SIMULATION PROGRAM.

voltage was made equal to 339.4 volts. The power dissipated by the load was 4200 (watts) + j280 (vars) under matched conditions - the same as in the GPC tests.

For a balanced condition with a capacitive load, R_L = 13.7 ohm and C_L = 12.895 μF. Table I-1 shows the sensitivity of the run-on time on load conditions as the load power is varied for fixed vars and fixed watts, respectively. The balanced load case produces the longest run-on time of only 7.943 msec.

Table I-2 depicts the comparison between the simulated and the experimental results. All run-on times are less than 8 msec. The inverter shuts down almost instantaneously. The computer results are considered acceptable as they are well within the "window" of 200 msec, which is considered important at it may affect the automatic reconnection of the PV system to the utility grid.

Table I-3 shows a breakdown of the simulation results according to the type of the load. It may be observed that the inverter shuts down almost immediately for practically all situations except when the load phase angle is between 0° and 2°.

REFERENCES

1. O. Wasynczuk and P. C. Krause, "Computer Modeling of the American Power Conversion Corporation Photovoltaic Power Conditioning System," SANDIA Report SAND87-7006, March 1987.

TABLE I-1. SIMULATION RESULTS FOR CAPACITIVE LOAD ($\phi_L > 2°$).

Load Power	Load Impedance	V_{ac} Phase-Lag	Run-On Time
3720.93-j280.01	15.393-j1.158	4.302°	2.604 msec
4204.38-j280.01	13.64 -j0.908	3.8085°	7.943 msec
4600.64-j280.01	12.474-j0.759	3.482°	5.632 msec
4204.38-j325.72	13.618-j1.055	4.43°	0.326 msec
4024.38-j280.01	13.64 -j0.908	3.8085°	7.943 msec
4204.38-j147.66	13.683-j0.481	2.013°	5.664 msec

TABLE I-2. COMPARISON OF SIMULATED AND EXPERIMENTAL RESULTS.

Case	GPC Test Results	Computer Simulation Results
#I.A.1 Matched Power	Load Power 4160 (Watts) -280 (Vars) System Amps 17.28 (A) SPC Volts 244.8 (V) Run-On Time 0.006 (sec)	Load Power 4204.38 (Watts) -280.01 (Vars) System Amps 16.66 (A) SPC Volts 237.66 (V) Run-On Time 0.007943 (sec)
#I.A.2 10% Less	Load Power 3720 (Watts) -288 (Vars) System Amps 17.36 (A) SPC Volts 244.8 (V) Run-On Time 0.004 (sec)	Load Power 3720.93 (Watts) -280.01 (Vars) System Amps 16.66 (A) SPC Volts 270.18 (V) Run-On Time 0.002604 (sec)
#I.A.3 10% More	Load Power 4600 (Watts) -276 (Vars) System Amps 17.28 (A) SPC Volts 243 (V) Run-On Time 0.001 (sec)	Load Power 4600.64 (Watts) -280.01 (Vars) System Amps 16.66 (A) SPC Volts 223.45 (V) Run-On Time 0.005632 (sec)

TABLE I-3. COMPUTER RESULTS AND COMPARISONS FOR THE APCC INVERTER

Case 1. ($\phi_L > 2°$)	ϕ_L	Load Watts ; Vars	Run-On Time
R-L Load (Vac Leads Iac)	+3.953°	4199.73 ; 290.22	2.018 msec
R-C Load (Vac Lags Iac)	-4.302°	3720.93 ; -280.01	2.604 msec
	-3.809°	4204.38 ; -280.01	7.943 msec
	-3.482°	4600.64 ; -280.01	5.632 msec
	-4.430°	4204.38 ; -325.72	0.326 msec
	-2.013°	4204.38 ; -147.66	5.664 msec
Case 2. ($\phi_L < 2°$)	ϕ_L	Load Watts ; Vars	Run-On Time
R-C Load (Vac Lags Iac)	-1.776°	4204.38 ; -130.40	8.984 msec
	-0.592°	4204.38 ; -43.47	927.865 msec
Case 3. ($\phi_L = 2°$)	ϕ_L	Load Watts ; Vars	Run-On Time
R Load	0.000°	4201.31 ; 0.00	594.987 msec

APPENDIX I-1

SIMULATION PROGRAM AND TYPICAL COMPUTER RESULTS

GENERAL DESCRIPTION OF THE SIMULATION PROGRAMS

FOR THE TESLACO AND APCC SPC INVERTERS

1. APCC Simulation Programs: (a) APCC2RC.PAS

   (b) APCC2RL.PAS

   (c) APCC2R.PAS

   (a) APCC2RC.PAS - Parallel R-C Load Configuration

   (b) APCC2RL.PAS - Series R-L Load Configuration

   (c) APCC2R.PAS - Resistive Load Configuration.

2. TESLACO Simulation Program: Program 'TESLACO.PAS' uses RLC load configurations similar to those used in the Georgia Power Company Laboratory Tests.

3. The above programs are written in Turbo Pascal Version 3.0 for IBM PC/XT or PC/AT and depict graphical results for both the TESLACO and APCC SPC inverters. A graphics card is required to run this program such as the CGA or HGA, but the HGA (Hercules Graphics Adapter) is recommended due to its higher resolution. With a CGA card, only half of the screen can be shown.

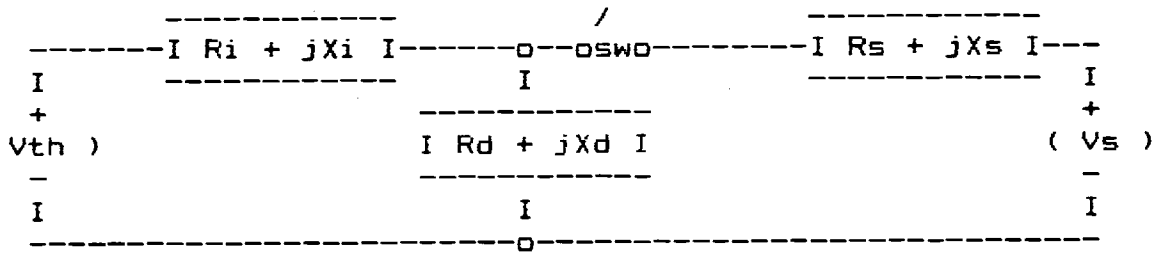
4. The APCC and TESLACO simulation programs read the input parameters from the keyboard and display the graphical results on the screen. After inputting the parameters, the simulation program shows the status of the initial conditions for the load impedance, the load power, etc. Then, pressing any key, the screen switches to the graphics mode and displays the results, i.e., all pertinent waveforms and other appropriate output information.

5. A 4th order Runge-Kutta integration subroutine is used to implement the state space models and the counters. In the TESLACO program, the 8-bit

counter is implemented by a 4-byte floating-point variable 'cnt' and the 9-bit counter in the APCC program is constructed by an integration subroutine.

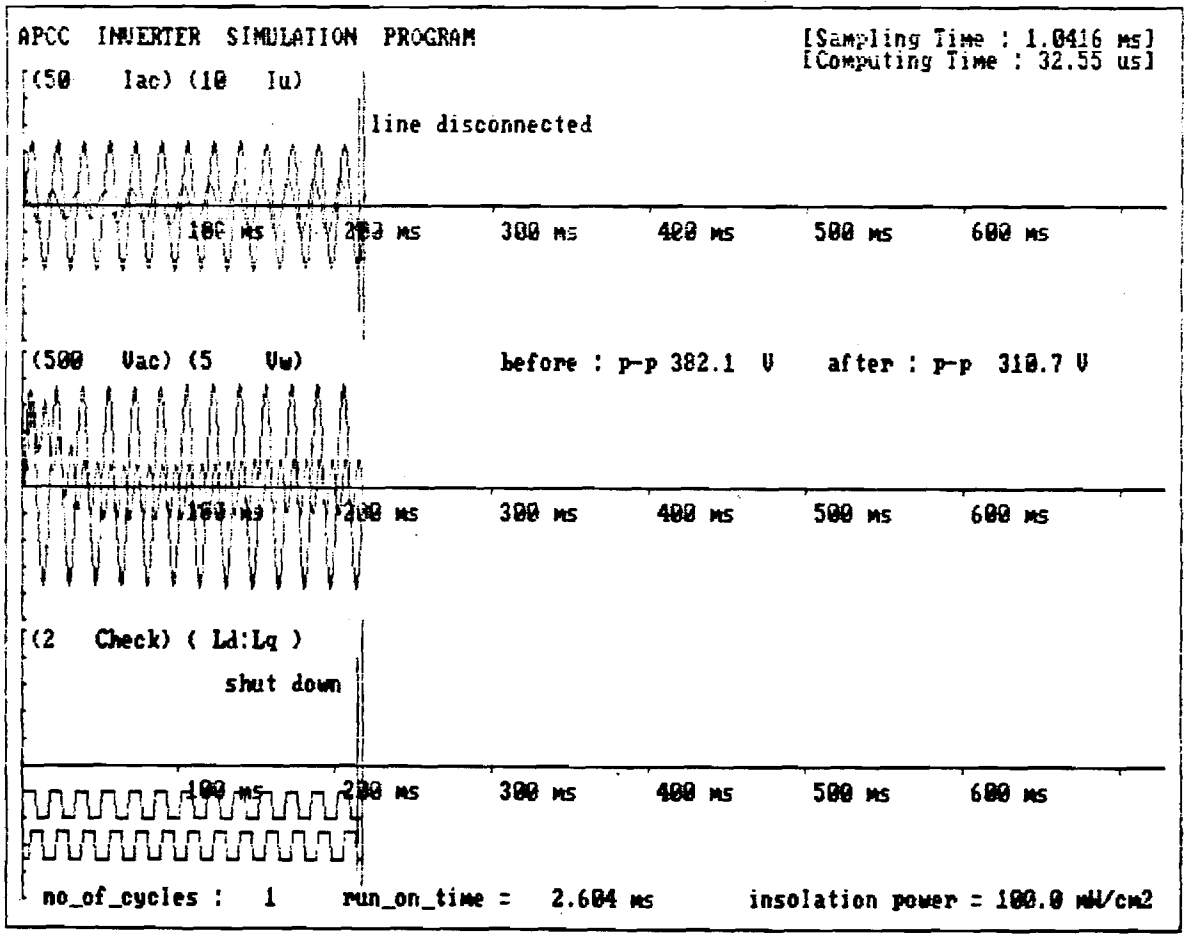
6. It takes a fairly long time to complete one simulation using an IBM PC/XT, but an IBM PC/AT 8 MHz is 5 times faster.

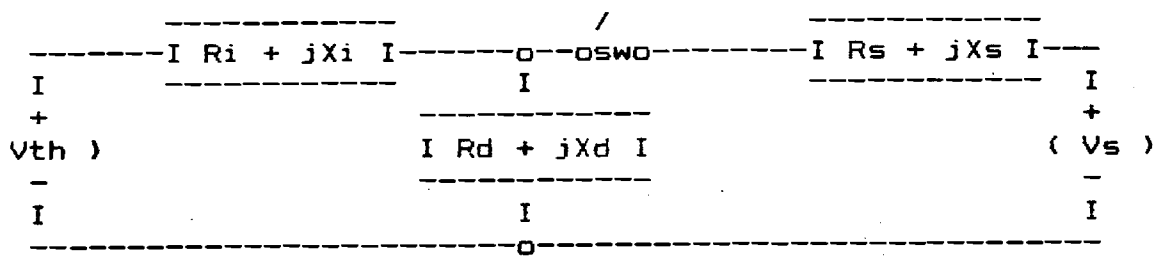
7. After shut-down, press the 'q' key to return to the text mode in the Turbo Pascal operating system; the graphics results remain on the screen. Use 'HPRINT.PAS' or 'HPRINT.COM' to hardcopy the results.




```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs         =    339.4                              [volts(p-p)]
Rd + jXd =     15.393     + j    -1.159             [ohms]
      ang =     -4.307 degree
Rs + jXs =      0.072     + j     0.072             [ohms]
Load Power =  3720.93     + j  -280.24              [Watts,Vars]
```

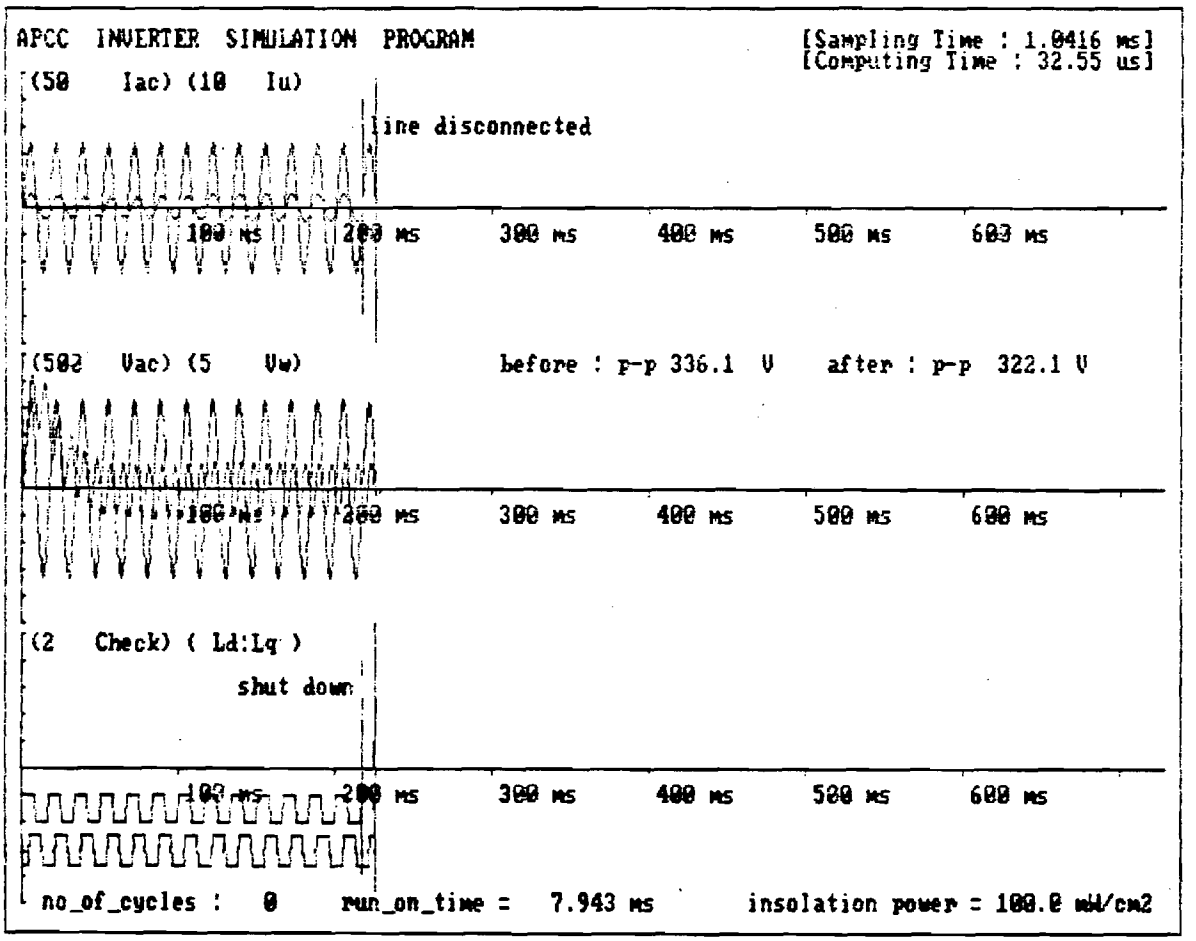
PRESS ANY KEY TO CONTINUE ...

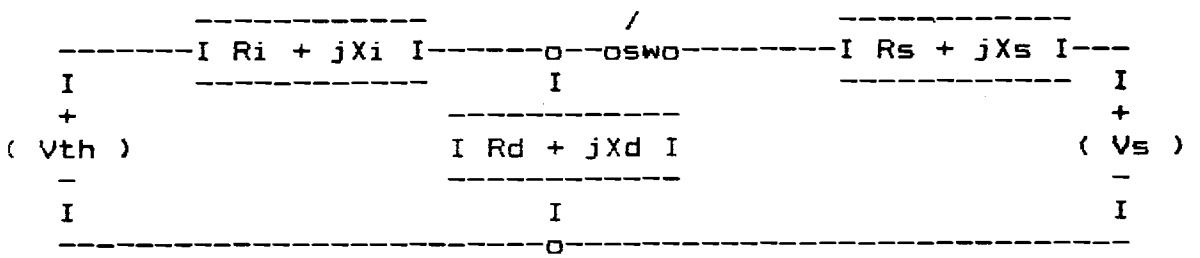




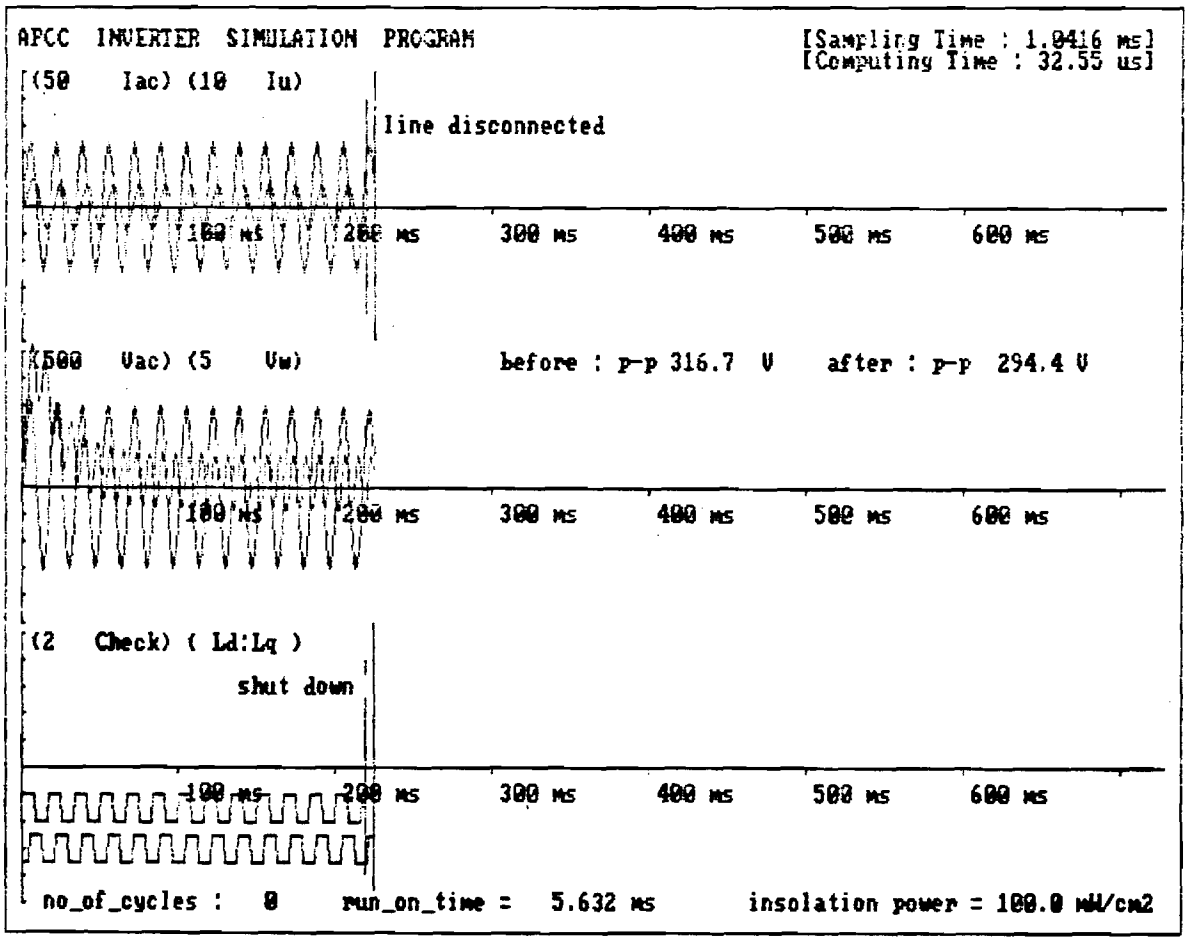

```
Pi = 100.0 [mW/cm2]     |Iref|   =    20.0000
Vs         =   339.4                          [volts(p-p)]
Rd + jXd   =    13.639     + j    -0.909       [ohms]
     ang   =    -3.813 degree
Rs + jXs   =     0.072     + j     0.072       [ohms]
Load Power =  4204.38     + j  -280.24         [Watts,Vars]
```

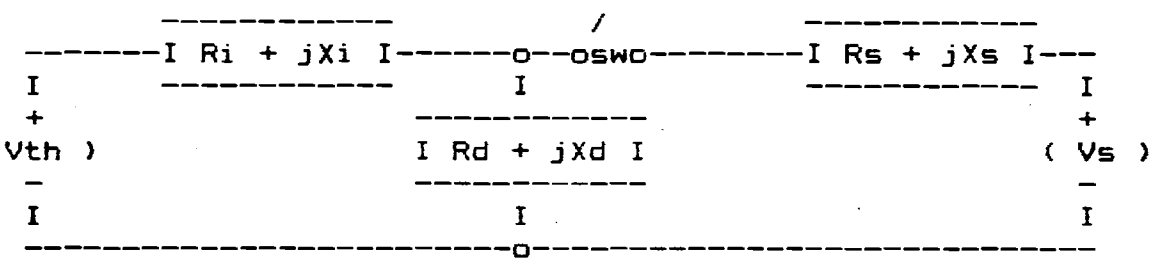
PRESS ANY KEY TO CONTINUE ...





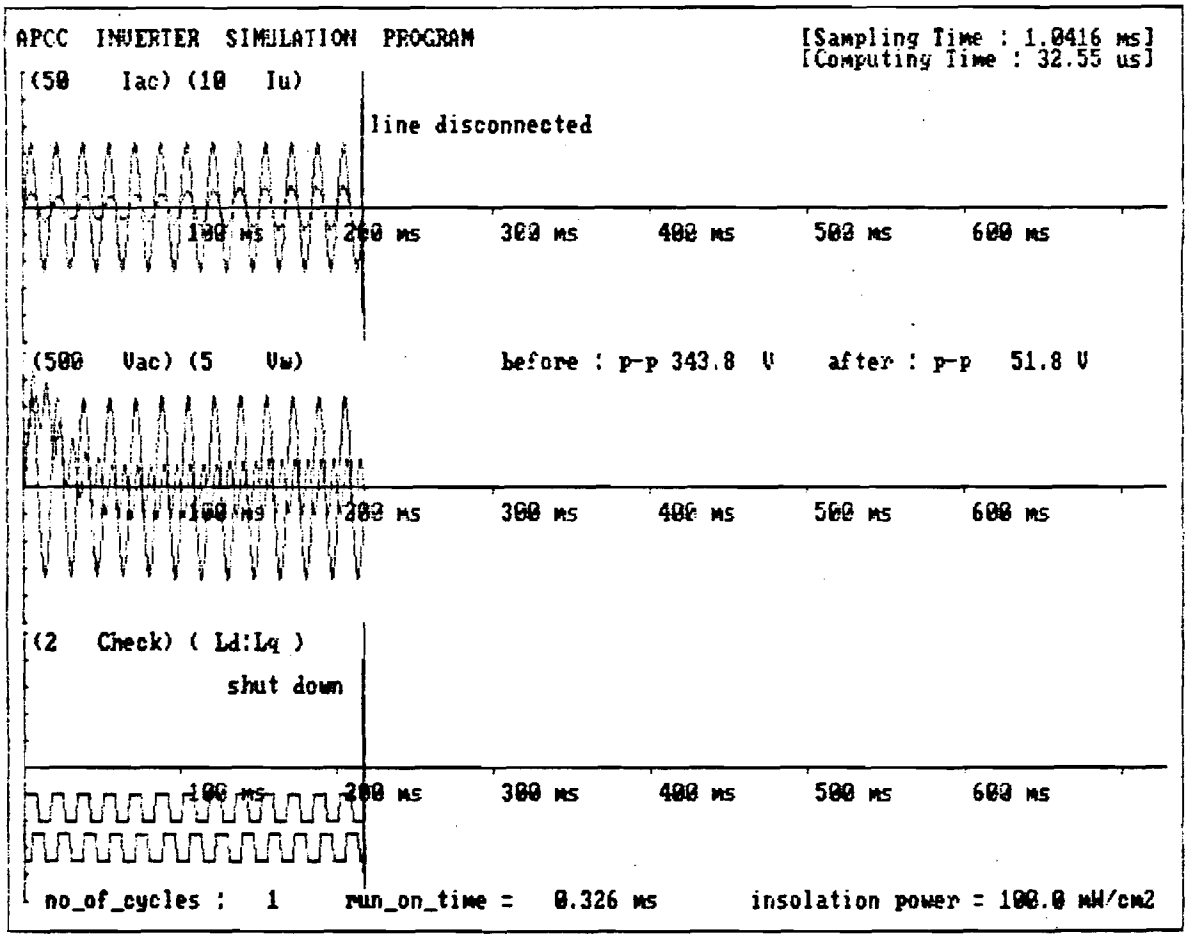

```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs         =    339.4                          [volts(p-p)]
Rd + jXd =     12.474     + j    -0.760         [ohms]
     ang =     -3.486 degree
Rs + jXs =      0.072     + j     0.072         [ohms]
Load Power =  4600.64     + j  -280.24          [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```

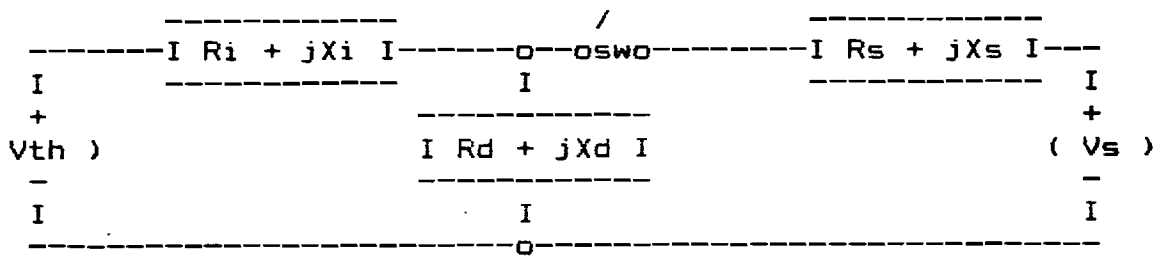




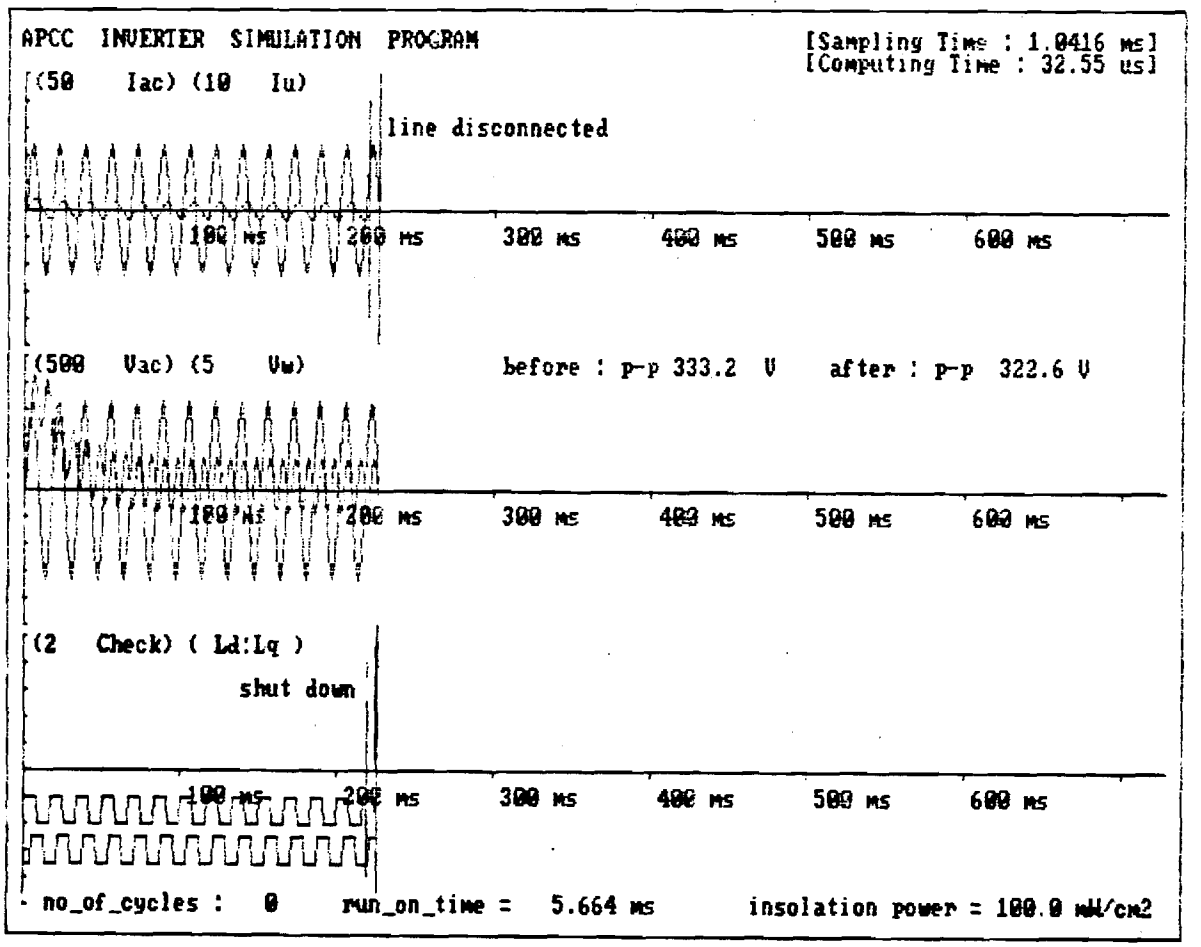

```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs         =    339.4                          [volts(p-p)]
Rd + jXd =     13.618     + j    -1.056         [ohms]
      ang =     -4.434 degree
Rs + jXs =      0.072     + j     0.072         [ohms]
Load Power =  4204.38     + j  -325.99          [Watts,Vars]
```

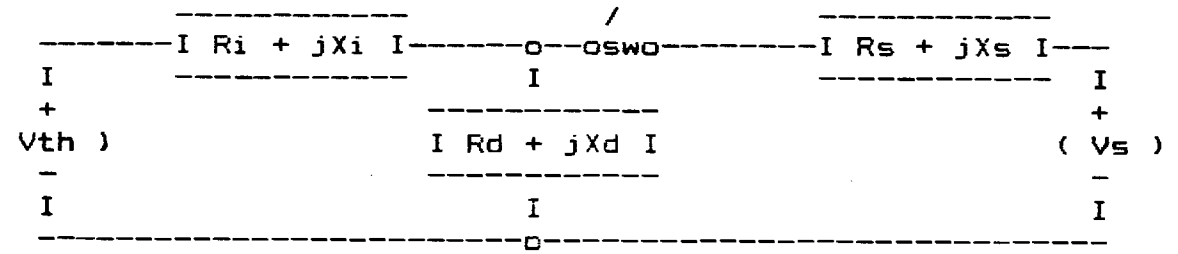
PRESS ANY KEY TO CONTINUE ...



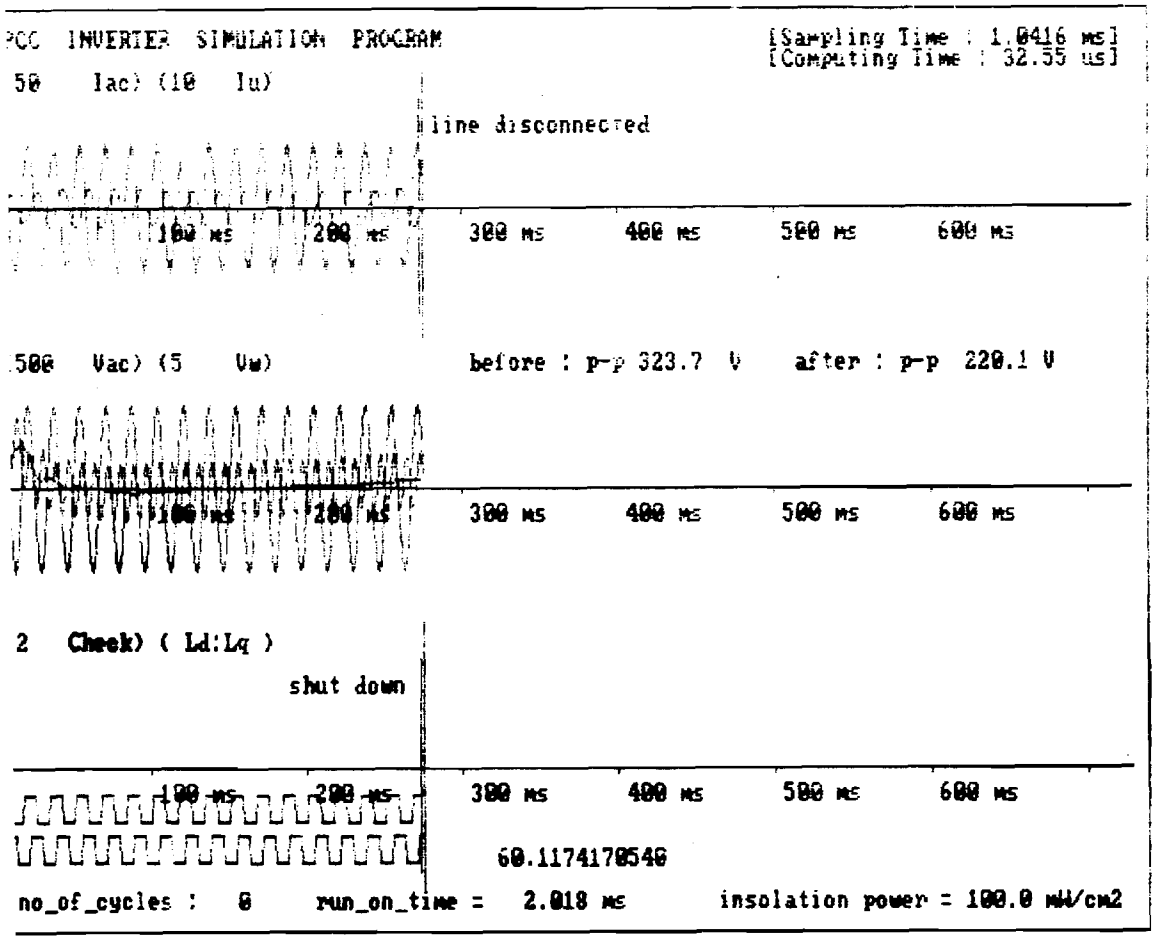


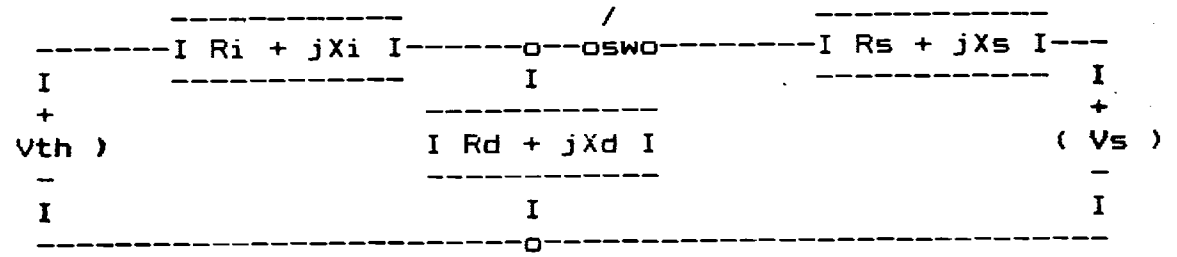

```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs         =   339.4                          [volts(p-p)]
Rd + jXd =     13.683     + j    -0.481        [ohms]
      ang =    -2.013 degree
Rs + jXs =      0.072     + j     0.072        [ohms]
Load Power =  4204.38     + j  -147.78         [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```



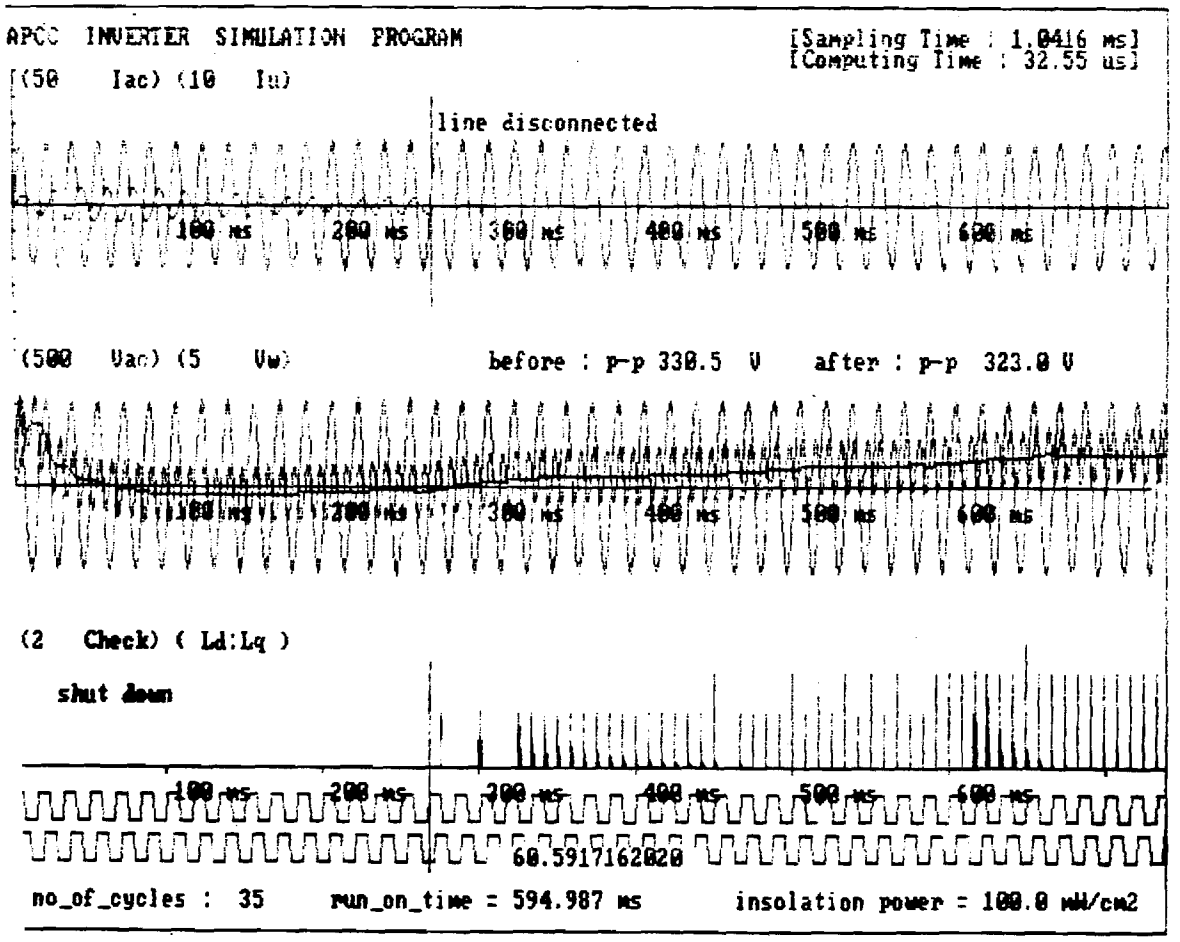


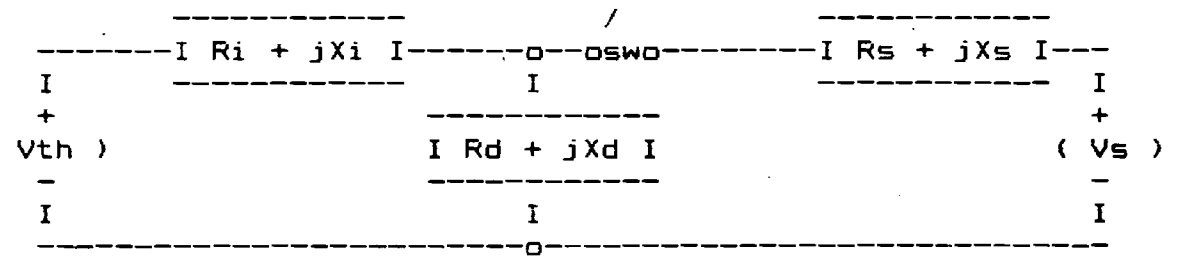

```
Pi = 100.0 [mW/cm2]     |Iref|   =    20.0000
Vs         =   339.4                          [volts(p-p)]
Rd + jXd   =    13.650     + j     0.943       [ohms]
     ang   =     3.953 degree
Rs + jXs   =     0.072     + j     0.072       [ohms]
Load Power =  4199.73      + j   290.22        [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```



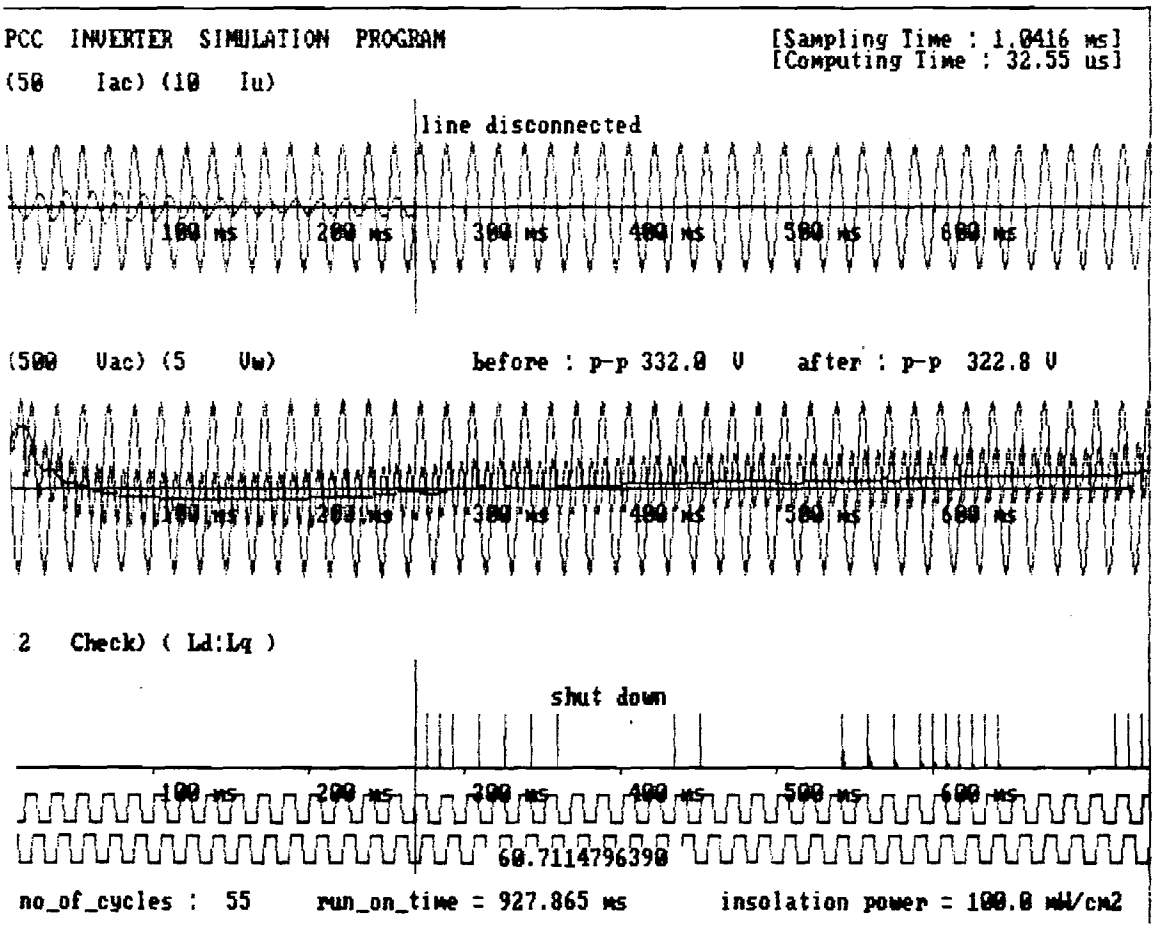


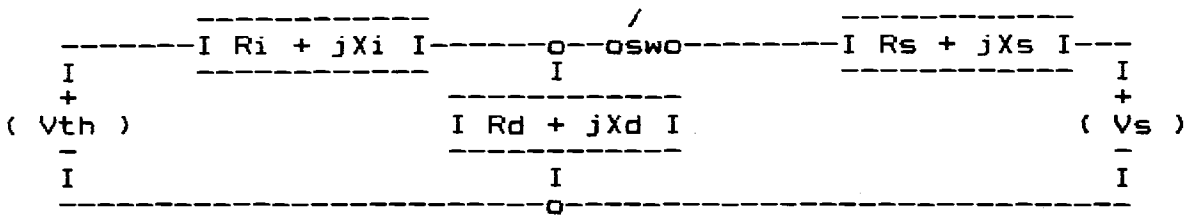

```
Pi = 100.0 [mW/cm2]     |Iref|   =    20.0000
Vs         =   339.4                          [volts(p-p)]
Rd + jXd =     13.710     + j     0.000        [ohms]
     ang =      0.000 degree
Rs + jXs =      0.072     + j     0.072        [ohms]
Load Power =  4201.31     + j      0.00        [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```



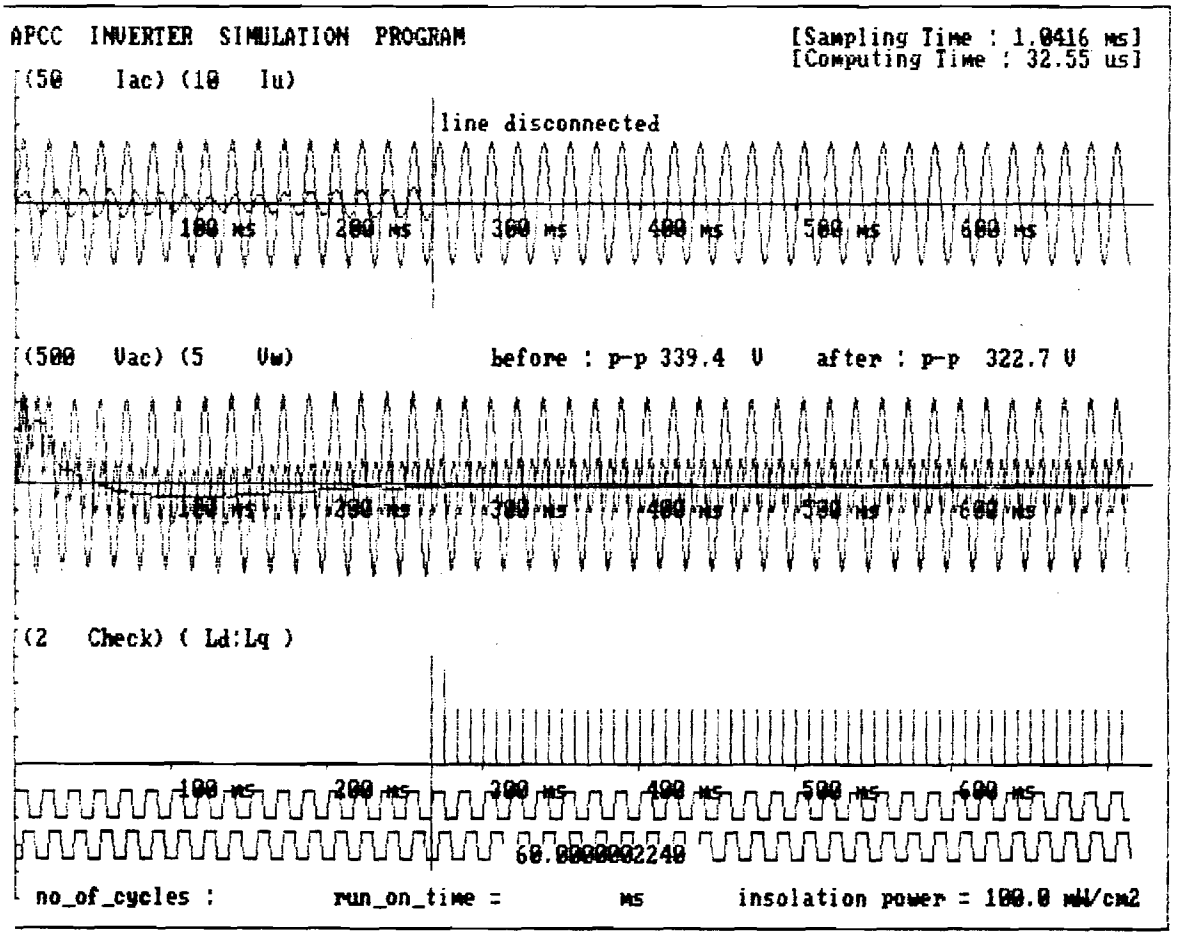



```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs         =   339.4                          [volts(p-p)]
Rd + jXd =     13.699     + j    -0.142        [ohms]
     ang =     -0.592 degree
Rs + jXs =      0.072     + j     0.072        [ohms]
Load Power =  4204.38    + j    -43.47        [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```






```
Pi = 100.0 [mW/cm2]     |Iref|    =    20.0000
Vs          =   339.4                              [volts(p-p)]
Rd + jXd =    13.696     + j    -0.248              [ohms]
     ang =    -1.036 degree
Rs + jXs =     0.072     + j     0.072              [ohms]
Load Power =  4204.38    + j    -76.06             [Watts,Vars]

PRESS ANY KEY TO CONTINUE ...
```



```
{ ***************************************************************************
  *                                                                         *
  *                  THIS PROGRAM SIMULATES THE ARCO INVERTER                *
  *                                                                         *
  *                  1987. 3.        Dr. George Vachtsevanos                *
  *                                  G.P.A.      Hoon      Kang             *
  *                                                                         *
  *                                  School of Electrical Engineering       *
  *                                  Georgia Tech.                          *
  *                                                                         *
  ***************************************************************************

PROGRAM Appic_3 output);

{ $graph_ib.BGI}

{$I Mandlop_.P+8}

CONST pi= 3.141592;
      steady_time= 6400;      {make it divided by 72}
      critical= 10;           {fine tuning}

TYPE  arrayss= ARRAY[1..3] of REAL;

VAR   h,v,e,x,y,v0,e0,x0,y0,v1,e1,x1,y1,
      v2,e2,x2,y2,v3,e3,x3,y3,v4,e4,x4,y4: arrayss;
      ex: CHAR; flag,shutdown: BOOLEAN;
      u,ux,t,x1,x2,x3,x4,b1,w0,
      Vw,Vt,Vac,Vack,Vu,Vv,
      ...,...,...,...,...,...,...,
      Tmpts,time,check,run_on_time: REAL;
      Lo,Lc,Lv,start_time,sw_neg,slope,sum: INTEGER;
      hw0,hx0,hy0,hz0,hu0,hv0,ns0,ht0,hw1,hx1,hy1,hz1,hu1,hv1,hs1,ht1,
      vw0,vx0,vy0,vz0,vu0,vv0,vs0,vt0,vw1,vx1,vy1,vz1,vu1,vv1,vs1,vt1: INTEGER;
      vrms,be_lcm,be_lcme,be_lcas,be_lcmp,be_lcap,af_lcm,af_lca,
      tx,phase_vs,phase_iref,vacmax,pi_power,xmean: REAL;
      s,r,q: ARRAY[1..4] OF INTEGER;
      fbuff: TEXT;
      buff: ARRAY[0..1000] OF REAL;
      npt,opt,crt: INTEGER;

PROCEDURE SetUp;
VAR  i: integer;
BEGIN
     ClrScr;
     repeat
          Gotoxy(10,10);
          write('Mono/Color Graphics Card [0=Mono/1=Color] ?');
          readln(crt);
     until crt in [0..1];
```

```
     ClrScr;
     FOR i:=1 TO 10 DO WRITELN;
     WRITE('                                   JUST A MOMENT ...');
     b[1]:= 1; b[2]:= 2; b[3]:= 2; b[4]:= 1;
     k1:= 0.5/15360; k2:= 8/15360;
     FOR i:= 1 TO 5 DO
     BEGIN
          x0[i]:= 0; x[i]:= 0; x1[i]:= 0; x2[i]:= 0; x3[i]:= 0; xA[i]:= 0;
     END;
     y1[1]:= 0.0;
     Vq:= 0; Vw:= 0; Lq:= 0; Ld:= 1; Lv:= 0; Theta:= 0;
     Vu:= 0; Iac:= 0; Vac:= 0; Vacx:= 0;
     check:= 0; itau:= 1/(0.1E-3);
     Imag:= 20; ex:= ' '; flag:= false; shutdown:= false;
     ux:= 0; u:= 0; t:= 0; run_on_time:= 0; sum:= 0; slope:= 1;
     w0:= 60.05*2*pi; { 60 Hz } phase_vs:= 0; phase_iref:= 0; vacrms:= 0;
     FOR i:= 1 TO 4 DO BEGIN s[i]:= 0; r[i]:= 1; q[i]:= 0; END;
     t:= 0;
     ASSIGN(fbuff,'freq.dat');
     REWRITE(fbuff);
     FOR i:= 0 TO 1000 DO buff[i]:= 0;
     rpt:= 256; npt:= 1; xmean:= 0;
END; {GetJg}


PROCEDURE RungeKutta4(VAR e,y,x,y: arrays; order: INTEGER; VAR p: INTEGER);
VAR  i,j,k: integer;
BEGIN
     CASE p of 1: for i:= 1 to order do
                  begin
                       e[i]:= k1*v[i]; y[i]:= v[i]; x[i]:= v[i] + 0.5*e[i];
                  end;
               2: for j:= 1 to order do
                  begin
                       x[j]:= k1*v[j]; e[j]:= e[j] + 2*b[p]*x[j];
                       x[j]:= y[j] + b1*x[j];
                  end;
               3: for j:= 1 to order do
                  begin
                       x[j]:= k1*v[j]; e[j]:= e[j] + 2*b[p]*x[j];
                       x[j]:= y[j] + b1*x[j];
                  end;
               4: for k:= 1 to order do
                  begin
                       x[k]:= e[k] + k1*v[k]; x[k]:= y[k] + x[k]/6;
                       exit;
                  end;
     end; {case}
     if p = 2 then b1:= 1;
     if p <> 2 then t:= t + 0.5*k1;
END; {RungeKutta4}
```

```
PROCEDURE LineFilter;
{ input: Vac      output: Vacx }
VAR  index,p: integer;
BEGIN
     b1:= 0.5;
     IF u <> 0 THEN
     BEGIN
          FOR p:= 1 TO 4 DO
          BEGIN
               v[1]:= -70970.59*x[1]-2303.92*x[2]+1470.59*x[3]+[illegible]
               v[2]:= -104.7237*x[1]-104.7237*x[2]+66.8449*x[3];
               v[3]:= 1470.59*x[1]+1470.59*x[2]-2597.43*x[3]+74.[illegible]*x[4]
                    -4925.7o*x[5];
               v[4]:= 46.39941*x[3]-46.39941*x[4]+3078.60*x[5];
               v[5]:= 46.39941*x[3]-46.39941*x[4]-46.39941*x[5];
               RungeKutta4(e,v,x,y,5,p);       { order = 5 }
          END;
          t:= t - k1;
     END;
     Vacx:= x[3];
END; {LineFilter}


PROCEDURE PhaseComparator;
{ inputs: Vacx,Lq  output: Vq }
VAR  XorOut,AD_Vacx: INTEGER;
BEGIN
     IF (Vacx>=0) THEN AD_Vacx:= 1
                  ELSE AD_Vacx:= 0;
     XorOut:= AD_Vacx XOR Lq;
     Vq:= ROUND(XorOut)*15 - 7.5;
END; {PhaseComparator}


PROCEDURE LoopFilter;
{ input: Vq       output: Vw }
VAR  p: integer;
BEGIN
     b1:= 0.5;
     IF u <> 0 THEN
     BEGIN
          FOR p:= 1 TO 4 DO
          BEGIN
               v1[1]:= x1[2];
               v1[2]:= x1[3]+6314.711571*Vq;
               v1[3]:= -18963.09781*x1[2]-365.608526*x1[3]-2289749.291*Vq;
               RungeKutta4(e1,v1,x1,y1,3,p);       { order = 3 }
          END;
          t:= t - k1;
     END;
     Vw:= 10.64*x1[1];
     buff[npt]:= Vw;
     xmean:= xmean + (buff[npt] - buff[opt])/256;
     npt:= (npt + 1) MOD 1000; opt:= (opt + 1) MOD 1000;
     Vv:= xmean;
END; {LoopFilter}
```

```
PROCEDURE Vcc_Counter;
VAR   p: INTEGER; dw: REAL;
      stx: STRING[20];
BEGIN
      dw:= 4*0.5829*Vv+w0;

      b1:= 0.5;
      IF u <> 0 THEN
      BEGIN
            FOR p:= 1 TO 4 DO
            BEGIN
                  v2[1]:= dw;
                  RungeKutta4(e2,v2,x2,y2,1,p);       { order = 1 }
            END;
            t:= t - k1;
      END;

      Ire:= Imag*ADC(SIN(x2[1]));

      Theta:= x2[1]*180/pi;
      IF Theta>=90 THEN slope:= -1;
      IF Theta<0 THEN slope:= 1;
      IF slope=1 THEN Lq:= 1 ELSE Lq:= 0;
      IF (Theta>=0) AND (Vacx>=0) THEN Ld:= 1 ELSE Ld:= 0;

      IF Theta>=360 THEN BEGIN
                              x2[1]:= 0;
                              Theta:= 0;
                              IF flag=True THEN sum:= sum + 1;
                              WRITELN(fbuff,1/(t-tx):15:10);
                              STR(1/(t-tx):15:10,stx);
                              Hgrwrite(stx,300,312,crt);
                              tx:= t;
                        END;
      IF (Theta>=0) AND (Theta<180) THEN Ld:= 1 ELSE Ld:= 0;
      IF (Theta>=90) AND (Theta<270) THEN Lq:= 1 ELSE Lq:= 0;
END; {Vcc_Counter}


PROCEDURE PhaseDetector;
VAR   p: integer; Lv,Lx: INTEGER;
      st: strings; Li: REAL;
BEGIN
      IF flag=True THEN
      BEGIN
            IF Vacx>=0 THEN Lv:= 1 ELSE Lv:= 0;
            Lx:= Ld XOR Lv;
            IF Lx=1 THEN Li:= 15 ELSE Li:= 0;

            b1:= 0.5;
            if u <> 0 then
            begin
                  for p:= 1 to 4 do
                  begin
                        v0[1]:= itau*(Li-x0[1]);
                        RungeKutta4(e0,v0,x0,y0,1,p); { order = 1 }
                  end;
```

```
            t:= t - k1;
        end;
        Check:= x0[1];
    end;
    if (round(ux) mod 4)=0 then
    begin
        Str(Check:7:3,st); hgrwrite(st,300,320,crt);
        Str(Vw:7:3,st); hgrwrite(st,300,328,crt);
        Str(vac.max:8:1,s0);
        if flag=false then hgrwrite(st,460,128,crt)
                      else hgrwrite(st,500,128,crt);
    end;
    if shutdown=false then
        if (abs(check)>critical) and (u>steady_time) then
        begin
            shutdown:= true;
            run_on_time:= (u - start_time)*k1*1000;
        end;
END; {PhaseDetector}


PROCEDURE PhaseLockedLoop;
BEGIN
    LineFilter;
    PhaseComparator;
    LoopFilter;
    Vcc_Counter;
    PhaseDetector;
END; {PhaseLockedLoop}


PROCEDURE DownConv_Unwrapper;
{ input: Iref      output: Iac }
VAR  Iacx: REAL;
BEGIN
    Iacx:= 0.5*(2.3bx*Iref);
    IF Ld=1 THEN Iac:= Iacx
            ELSE Iac:= -Iacx;
END; {DownConv_Unwrapper}


PROCEDURE SimulationModel;
  input: Vac,Vu,IL      output: Vac }
VAR  p: integer; temp: real;
BEGIN
    temp:= 120*pi*t;
    Vu:= Vrms*sin(temp);
    IF flag=False THEN          { before the line_disconnection }
    BEGIN
        t1:= 0.5;
        IF u <> 0 THEN
        BEGIN
            FOR p:= 1 TO 4 DO
            BEGIN
                v3[1]:= (Vu-Vac-0.072*x3[1])/0.072;
                RungeKutta4(e3,v3,x3,y3,1,p);      { order = 1 }
            END;
            t:= t - k1;
        END;
        Iuo:= Iu;
        Iu:= x3[1];
```

```
        IL:= Iu + Iac;
        IF (u>steady_time+1500) AND (Iu>=0) AND (Iuo<0) THEN
        BEGIN
              flag:= True; start_time:= round(ux);
              linex(hx1,74-40,hx1,74+40,crt);
              linex(hx1,282-40,hx1,282+40,crt);
              Hgrwrite('line disconnected',hx1+3,40,crt);
              vacmax:=0;
        END;
    END;
    IF flag=True THEN              { after the line_disconnection }
    BEGIN
        Iu:= 0;
        IL:= Iac;
    END;
    b1:= 0.5;
    IF u <> 0 THEN
    BEGIN
        FOR p:= 1 TO 4 DO
        BEGIN
            v4[1]:= (-x4[1]*IRL-IL)*IOL;
            RungeKutta4(e4,v4,x4,y4,1,p);       { order = 1 }
        END;
        t:= t - k1;
    END;
    Vac:= x4[1];
    IF (abs(vac)>vacmax) THEN vacmax:= abs(vac);
    t:= t + k1;
END; {SimulationModel}


PROCEDURE GraphInit;
BEGIN
    InitGrf(crt);
    GMode(crt);
    ClearScr(crt);
    Linex(0,0,719,0,crt); Linex(719,0,719,343,crt);
    Linex(719,343,0,343,crt); Linex(0,343,0,0,crt);
    Hgrwrite
    ('APCC  INVERTER  SIMULATION  PROGRAM',6,8,crt);
    Hgrwrite('[Sampling Time : 1.0416 ms]',480,8,crt);
    Hgrwrite('[Computing Time : 32.55 us]',480,16,crt);
    Hgrwrite('insolation power =       mW/cm2',450,328,crt);
    Str(pi_power:5:1,st); Hgrwrite(st,600,328,crt);
    Hgrwrite('#_of_steps = ',20,328,crt);
    Frame('(50    Iac) (10   Iu)',10,74,crt);
    Frame('(500   Vac) (5    Vw)',10,178,crt);
    Frame('(2 Check) ( Ld:Lq )',10,282,crt);
    Hgrwrite('before : p-p        V',300,128,crt);
    Hgrwrite('after : p-p        V',500,128,crt);
    hw1:= 10; hx1:= 10; hy1:= 10;  hz1:= 10;
    hu1:= 10; hv1:= 10; hs1:= 10;  ht1:= 10;
    vw1:= 74-ROUND(Iac);       vx1:= 74-ROUND(Iu*5);
    vy1:= 178-ROUND(Vac*0.1); vz1:= 178-ROUND(Vw*10);
    vu1:= 317-ROUND(Lq*10);    vs1:= 302-ROUND(Ld*10);
    vt1:= 282-ROUND(Check*5); vv1:= 178-ROUND(Vv*10);
END; {GraphInit}
```

```
PROCEDURE GraphResult;
BEGIN
     vw0:= vw1; vw1:= 74-ROUND(Iac); hw0:= hw1; hw1:= hw1+1;
     Linex(hw0,vw0,hw1,vw1,crt);
     vx0:= vx1; vx1:= 74-ROUND(Iu*5); hx0:= hx1; hx1:= hx1+1;
     Linex(hx0,vx0,hx1,vx1,crt);

     vy0:= vy1; vy1:= 178-ROUND(Vac*0.1); hy0:= hy1; hy1:= hy1+1;
     Linex(hy0,vy0,hy1,vy1,crt);
     vz0:= vz1; vz1:= 178-ROUND(Vw*10); hz0:= hz1; hz1:= hz1+1;
     Linex(hz0,vz0,hz1,vz1,crt);
     vv0:= vv1; vv1:= 178-ROUND(Vv*10); hv0:= hv1; hv1:= hv1+1;
     Linex(hv0,vv0,hv1,vv1,crt);

     vs0:= vs1; vs1:= 302-ROUND(Ld*10); hs0:= hs1; hs1:= hs1+1;
     Linex(hs0,vs0,hs1,vs1,crt);
     vu0:= vu1; vu1:= 317-ROUND(Lq*10); hu0:= hu1; hu1:= hu1+1;
     Linex(hu0,vu0,hu1,vu1,crt);
     vt0:= vt1; vt1:= 282-ROUND(Check*5); ht0:= ht1; ht1:= ht1+1;
     Linex(ht0,vt0,ht1,vt1,crt);
END; {GraphResult}


PROCEDURE PhasorInit;
VAR  ri,xi,rd,xd,rs,xs,tempr,pva_power,vpva,arg,x,y,z: REAL;

     PROCEDURE complex_mult(x1,x2,y1,y2: real; var z1,z2: real);
     BEGIN
          z1:= x1*y1 - x2*y2;
          z2:= x1*y2 + x2*y1;
     END;

     PROCEDURE complex_div(x1,x2,y1,y2: real; var z1,z2: real);
     VAR  temp: real;
     BEGIN
          temp:= y1*y1 + y2*y2;
          if temp=0 then begin
                              writeln('divided by zero error');
                              exit;
                         end;
          z1:= x1*y1 + x2*y2; z1:= z1/temp;
          z2:= x2*y1 - x1*y2; z2:= z2/temp;
     END;

     FUNCTION f(x: real): real;
     BEGIN
       f:= 1.7533E-4*x*(exp(x/21.26)-1)-22.12*x*(pi_power/100)+pva_power;
     END;

     FUNCTION df(x: real): real;
     BEGIN
       df:=1.7533E-4*(1+x/21.26)*exp(x/21.26)-22.12*(pi_power/100)-1.7533E-4;
     END;
```

```
    PROCEDURE find_load(x,y: real);
    BEGIN
         complex_div(1,0,1/x,(w0*y*1.0E-6),rd,xd);
    END; {find_load}

BEGIN
    ClrScr;
    rs:= 0.072; xs:= 0.072; rd:= 13.7; xd:= -0.91333; vrms:= 240;
    writeln('input your line impedance (Rs+jXs)');
    write  ('       default = 0.072 + j 0.072 [ohm] : ');
    readln (rs,xs);
    x:= 13.70; y:= 12.895;
    writeln('input your load impedance (R  C[uF]) : ');
    write  ('       default = 13.70 [ohms] // 12.895 [uF] : '); readln(x,y);
    IOL:= 1/(y*1.0E-6); IRL:= 1/x;
    find_load(x,y);
    writeln('input your line voltage Vs in RMS ');
    write  ('       default = 240.0              [rms] : ');
    readln (vrms);
    vrms:= vrms*Sqrt(2); pi_power:= 100; pva_power:= 4200;
    writeln('input your insolation PI (mW/cm^2)');
    write  ('       default = 100.0          [mW/cm^2] : ');
    readln (pi_power);
    writeln('input your photovoltaic array Ppva');
    write  ('       default = 4200                [W] : ');
    readln (pva_power);
    tempr:= 0.5*SQR(vrms)*SQRT(SQR(IRL)+SQR(w0*y*1.0E-6));
    arg:= ArcTan(-w0*y*1.0E-6*x);

    ClrScr;
    writeln;
    writeln('           ------------          /          ------------          ');
    writeln('    -------I Ri + jXi I------o--oswo---------I Rs + jXs I---     ');
    writeln('    I      ------------      I               ------------  I      ');
    writeln('    +                   ------------                      -      ');
    writeln('  ( Vth )                 I Rd + jXd I                     )     ');
    writeln('    -                   ------------                      -      ');
    writeln('    I                        I                            I      ');
    writeln('    -------------------------o-----------------------------       ');
    writeln; writeln;
    writeln('    Pi = ',pi_power:5:1,' [mW/cm2]','     !Iref! =  ',Imag:5:1,
    writeln('    Vs       =  ',vrms:6:1,
                                                    [volts(p-p)] );
    writeln('    Rd + jXd =  ',rd:6:3,'    + j ',xd:8:3,'          [ohms] );
    writeln('         ang =  ',arctan(xd/rd)*180/pi:8:3,' degree');
    writeln('    Rs + jXs =  ',rs:8:3,'    + j ',xs:8:3,'          [ohms]');

    writeln('    Load Power = ',tempr*cos(arg):8:2,
            '    + j ',tempr*sin(arg):8:2,'          [Watts,Vars]');
    writeln; writeln;
    write  ('    PRESS ANY KEY TO CONTINUE ... '); readln;
END; {PhasorInit}


 main program }


BEGIN
    SetUp;
    PhasorInit;
    GraphInit;
```

```
    WHILE (u<steady_time+70000.0) AND (ex<>'q') AND (ex<>'p') DO
    BEGIN
         PhaseLockedLoop;
         DownConv_Unwrapper;
         SimulationModel;
(*       IF (u>=steady_time) AND (u<steady_time+21504) AND
            (round(ux) mod 32 = 0) THEN GraphResult;          *)
         IF (ROUND(ux) MOD 32 = 0) THEN GraphResult;
         Linex(ht1,292,ht1,292-ROUND(Check*5),crt);
         u:= u + 1.00; ux:= ux + 1.00; if ux>=16385 THEN ux:= 1;
         Str(u:5:0,st); Hgrwrite(st,120,328,crt);
         IF shutdown=true THEN
         BEGIN
              IF u<steady_time+21504 THEN
              BEGIN
                   Linex(hx1,282-50,hx1,282+50,crt);
                   Linex(hx1,74-50,hx1,74+50,crt);
              END;
              Hgrwrite('shut down',hx1-88,248,crt);
              ex:= 'q';
         END;
         IF KEYPRESSED THEN READ(Kod,ex);
    END;
    Hgrwrite('no_of_cycles : ',20,328,crt);
    Str(sum:4,st); Hgrwrite(st,130,328,crt);
    Str(run_on_time:7:3,st); Hgrwrite('run_on_time = '+st+' ms',200,328,crt);
    st:= '                    '; Hgrwrite(st,300,320,crt);
    IF ex='p' THEN HardCopy;
    READLN;
    Tmode(crt);
    READLN;
    CLOSE(fbuff);
END. { main program }
```

PART II

A SIMPLIFIED ANALYTICAL MODEL FOR THE TESLACO 4 kW PHOTOVOLTAIC INVERTER

A utility-interactive self-commutated, voltage-sourced inverter based on high frequency isolation has been developed by TESLACO to provide a photovoltaic to utility interface at the residential power level of 4 kW. Figure II-1 is a block diagram of the TESLACO T-4 kW-A power stage and control circuitry. A detailed description of the TESLACO inverter is given in the excellent paper by A. Cocconi et al. [1]. The development of the simplified model was based primarily on information contained in this paper as well as the hybrid computer simulation conducted at Purdue University [2].

II.1 General Description

Refer to the block diagram of Figure II-1. The DC power from the solar array is converted to a full wave rectified sinusoidal waveform through the push-pull buck stage converter. The buck stage also provides isolation between the DC and AC sides of the circuit through the use of a 20 kHz compact and lightweight transformer. In the second stage of the inverter, alternate half sine waves are inverted by unfolding to form the complete output sine wave that is matched to the 240 volt AC line.

The control circuitry performs a multiplicity of functions ranging from tracking the maximum power point of the PV array to generating the PWM base drive signals and controlling the output unwrapper transistor bridge. A brief overview of the control strategy follows, emphasizing those features that are relevant to an islanded operation of the inverter. The peak power tracker

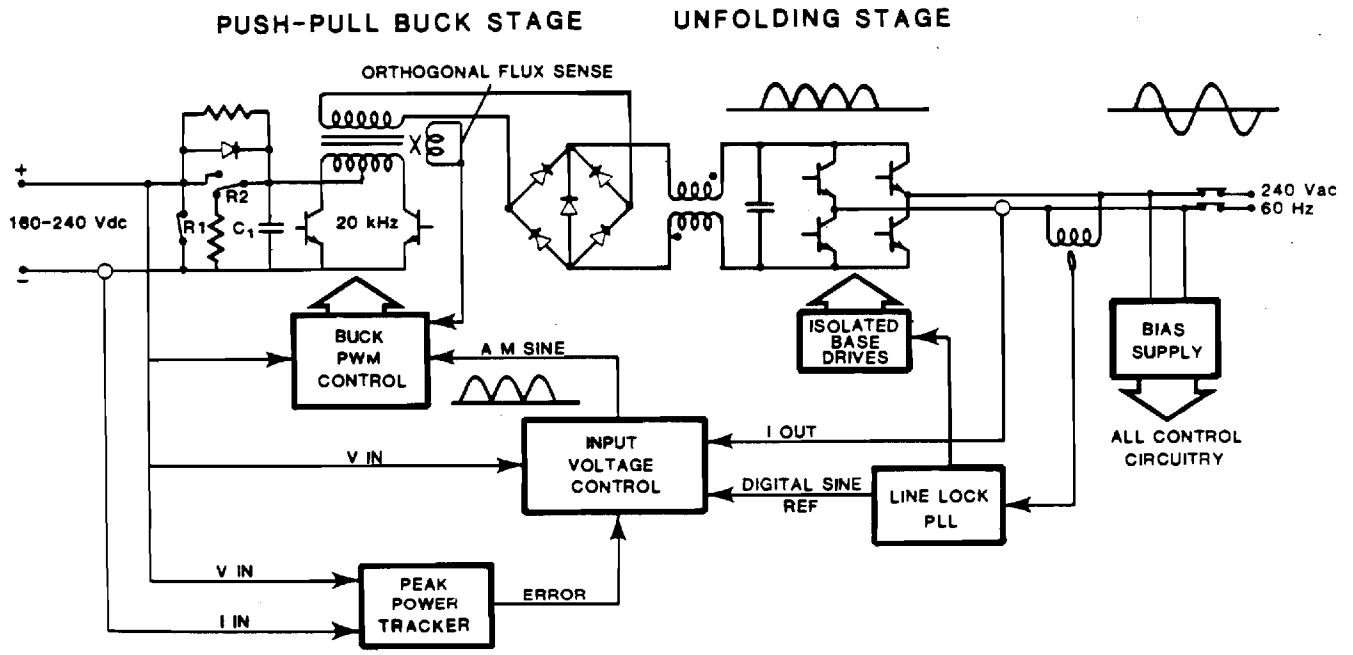


FIGURE II-1. BLOCK DIAGRAM OF THE TESLACO T-4 kW-A POWER STAGE AND CONTROL CIRCUITRY.

follows the array peak-power point by continually monitoring the ripple on the DC input. Its operation is based upon balancing the symmetry of the 120 Hz power ripple signal. The unfolding transistor control circuitry monitors the current in each of the four transistors and acts to turn them off rapidly in the event of a fault condition on the line. The output current feedback control circuit measures the average AC output current of the system and modulates the signal to the buck control circuit so as to maintain it at the desired level.

The most crucial control component responsible for shutting down the transistor bridge in the event of a utility disconnect is the line lock phase-locked loop (PLL). A reference rectified sine wave voltage is generated internally and locked to the line by use of the digital PLL circuitry. Current delivered to the line is automatically in phase with the voltage, once it is brought into synchronization with the line through the PLL circuitry. A phase discrepancy between the line and reference signal is used to destabilize the loop upon line disconnection resulting ultimately in inverter shutdown.

II-2. Model Development

It is assumed that the PV system is in a steady state condition before line disconnection. The proposed steady state equivalent circuit is shown in Figure II-2.

where

$Zi = 6.27 + j0.28$

$Zs = 0.072 + j0.072$

$Vi = 390 * Iref \angle 0°$

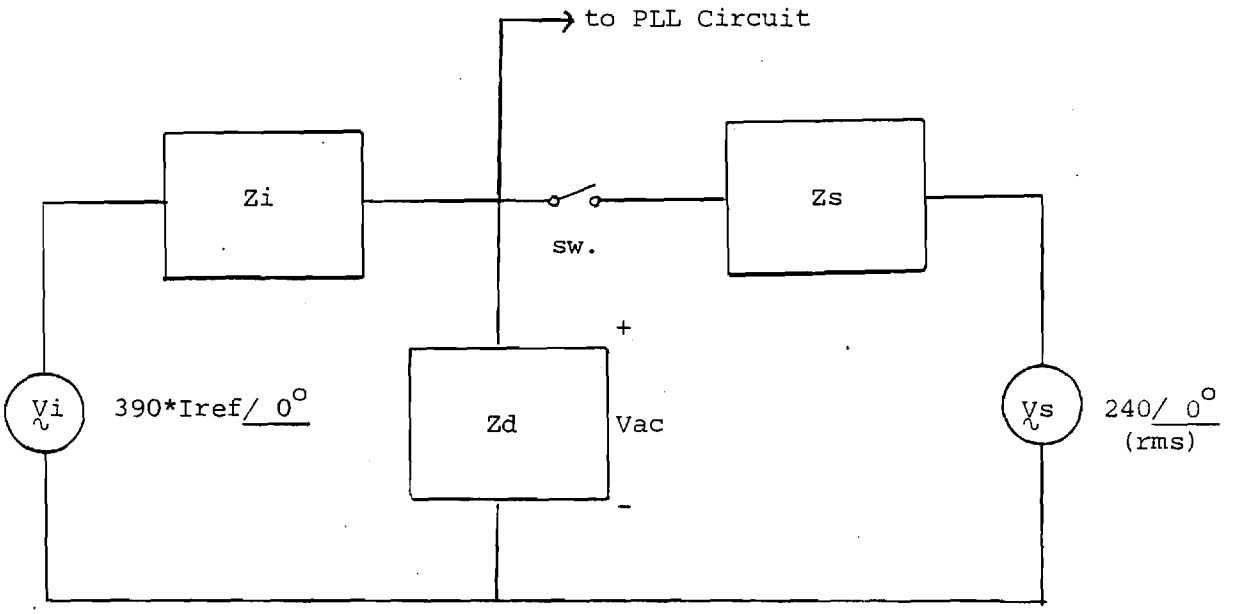


FIGURE II-2. STEADY STATE EQUIVALENT OF THE INVERTER-LOAD-UTILITY INTERCONNECTION

$Vs = 339.4\angle 0°$

$V_{ac} = L_1(Vi) + L_2(Vs)$, L_1 and L_2 are linear functions

Zd = the load impedance.

The equivalent circuit is effectively a "T-circuit" with a circuit breaker on the right branch. Since the inverter is simplified in terms of a steady state model, then a classical Thevenin-Norton equivalent network approach is appropriate although the device itself exhibits a nonlinear behavior. The balanced or matching condition is defined as follows:

The power delivered to the load should be the same before and after the line disconnection in order to prevent any instantaneous power difference.

In this matching condition, Vac,off is equal to Vac,on (p-p 339.4 V) and the phase difference between Vac,off and Vi is zero. If we define by $Zi = Ri + jXi$ and $Zd = Rd + jXd$, then the balanced condition can be derived as:

$$Vac,off = \frac{(Rd + jXd)*Vi}{(Rd + Ri) + j(Xd + Xi)} \tag{1}$$

where Vac,off denotes the voltage across the load after line disconnection and Vi denotes the Thevenin equivalent voltage while $Zi = 6.27 + j0.28$. The power at the load becomes:

$$P_{load} = \frac{1}{2} V*\overline{I} = \frac{|Vac|^2}{2|Zd|^2} * Zd \tag{2}$$

At the balanced state, the real part of P_{load} is 4 kW and there is no phase difference between Vac,off and Vi:

From Eq. (2):

$$\frac{Xd}{Rd} = \frac{Xd + Xi}{Rd + Ri} = r = 0.0447 \tag{3}$$

i.e., $Re(P_{load}) = 0.5*339.4^2/Rd/(1 + r^2) = 4000$, therefore Rd = 14.362 for r = 0.0447 and Xd = 0.642. The sufficient condition for Eq. (3) to hold is that:

$$Xd/Rd = Xi/Ri = 0.0447$$

And the imaginary part of P_{load} is r*4000 = 178.8 (vars). Since the PLL plays a significant role in interrupting the power flow from the inverter to the utility grid under fault conditions, its operational characteristics will be described more fully in the next paragraph.

Figure II-3 shows a block diagram of the PLL circuit. The phase comparator is preceded by an integrator in order to smooth our fast disturbances on the AC line. The 90° phase shift of the integrator is compensated by decoding the 90° address of the EPROM clock. The loop filter is the only part of the circuit that introduces any dynamics into the modulation transfer function. The loop filter uses an ordinary lead-lag network for phase compensation and is represented by its transfer function. The output of the phase comparator, I_o, is equal to +0.227 mA (-0.227 mA) during the period of time when the reference pulse, R, leads (lags) the pulse, V, and is zero otherwise. Thus, during steady state operation at 60 Hz, R and V are in phase so that I_o is zero.

The output of the loop filter is described by the following relation:

$$\frac{V_y(s)}{I_o(s)} = \frac{1.47 \times 10^5}{s} + \frac{33000}{1 + s(7.26 \times 10^{-3})}$$

If the loop filter output, V_y, is zero, then the voltage control oscillator produces a pulse train of 15.36 kHz (60 Hz x 256). When V_y is positive

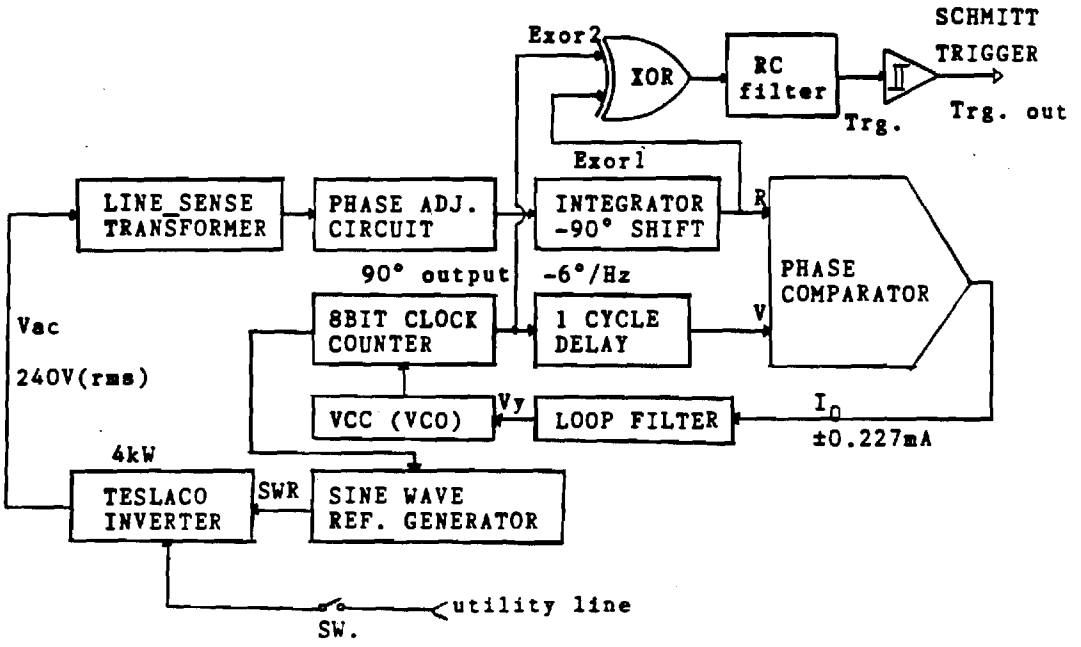


FIGURE II-3. BLOCK DIAGRAM OF THE LINE LOCKING PLL.

(negative), then the VCO produces a frequency deviation of Δf = (920 Hz) x (V_y in volts) with a corresponding increase (decrease) in the pulse train frequency. Thus, the comparator-filter combination acts as follows: If the reference signal R leads V, the output of the phase comparator becomes positive causing V_y to increase. This increases the VCO frequency which, in turn, causes the pulse V to occur earlier as desired. The opposite situation occurs if R lags V.

The 8-bit clock produces two output signals: The first one is the sine wave reference (stored in ROM) which, for all practical purposes, may be represented as a continuous signal by the expression

$$SWR(t) = 1|\sin[\varphi_{VCO}(t)]|$$

where

$$\varphi_{VCO}(t) = \frac{\text{count}}{256} \times 360^\circ$$

is the angular equivalent of the counter output whose output increments by 1 at each input clock pulse.

A second (÷ 256) output is provided which is equal to a logic "1" whenever $90^\circ \leq \varphi_{VCO}(t) \leq 270^\circ$. This second output is fed into a 1.00 cycle time delay. Although this 1.00 cycle time delay produces no locking angle error at exactly 60 Hz, it does introduce a transport lag in the control loop. It is exactly this 1.00 cycle transport delay that introduces a right half-plane pole of the loop gain function ensuring that the PLL will be unstable when the AC line is disconnected. As a matter of fact, the loop gain transfer function has a left half-plane pole without the 1.00 cycle time delay resulting in a stable system. The delay is, therefore, intentionally designed into the

system in order to bring about the unstable condition whenever R and V are out of phase. Finally, under free run conditions, the line sense transformer is essentially connected directly to the digital sine generator via the power stage of the inverter and a phase-adjust circuit is introduced to balance out phase contributions between the internal reference wave and the terminal voltage.

The PLL circuitry of the inverter detects when the line deviates from its nominal frequency or voltage by more than a preset value and then initiates the orderly inverter shutdown. In the case of an isolated or islanded condition, the inverter must be capable of detecting the absence of the line voltage in order to initiate the shutdown procedure. Consider the typical situation of a nonunity power factor load depicted in Figure II-4. Assume that the reactive load is partly supplied from the utility. Upon the line disconnection, the output voltage will be shifted in phase from the inverter's internal equivalent voltage source, thereby producing the phase shift shown in the figure. This phase discrepancy is detected by the PLL as a shift in frequency. If the shift in frequency is already beyond the preset limit (1.0 Hz or 6° phase shift), the inverter is forced to shut down. When the phase shift is initially insufficient to trigger the inverter shutdown, then the loop instability, as described above, assists in building up the phase difference until the critical condition is reached again. The time duration between the utility disconnect and the inverter shutdown is defined as the run-on time.

For a simplified representation of the PV system-utility grid under islanded conditions, attention must be focused, therefore, upon the dynamic

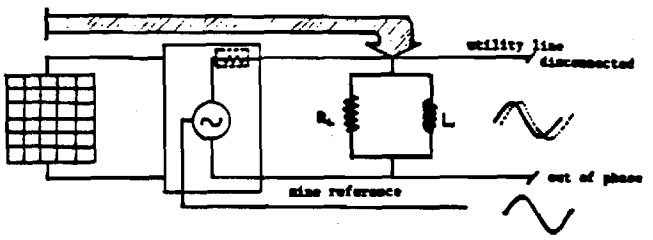


FIGURE II-4. EFFECT OF UTILITY LINE DISCONNECTION ON PHASE SHIFT.

behavior of the PLL circuitry responsible for the regenerative phase difference instability, on one hand, and the determination of the initial phase difference between the internal reference waveform and the terminal voltage upon disconnection of the utility line. Other major components of the power conditioning subsystem (power tracker, unfolder, etc.) do not seem to influence significantly the destabilizing behavior of the inverter. The initial phase difference is estimated approximately from a steady state representation of the buck stage-unfolder-load-utility line combination. Refer to Figure II-5(a). The first part of the circuit is a Thevenin equivalent representation of the buck power stage with voltage feedforward coupled with current feedback. The purpose of the current feedback is to provide the inverter with a nondissipative resistive output impedance of approximately 6.24 Ω. In addition to correcting for distortion in the AC line waveform, this output impedance is used to control the magnitude of the AC output current. For simplicity, the unfolder circuit is represented as a unity gain device. The remaining components in Figure II-5(a) represent filtering, the local loading, and a Thevenin equivalent of the utility line at the point of connection. The reference current, I_{ref}, may be varied by varying the input solar insolation to the PV array. Figure II-5(b) shows the equivalent steady state representation of II-5(a) after some simple manipulations.

Returning to the PLL circuit of Figure II-3, the simulation program is based on a representation of the continuous and discrete subsystems of the device. The 8-bit counter variable, cnt, (real) is updated at each computing time as follows:

$$cnt := cnt + 1 + \frac{920}{15360} Vy$$

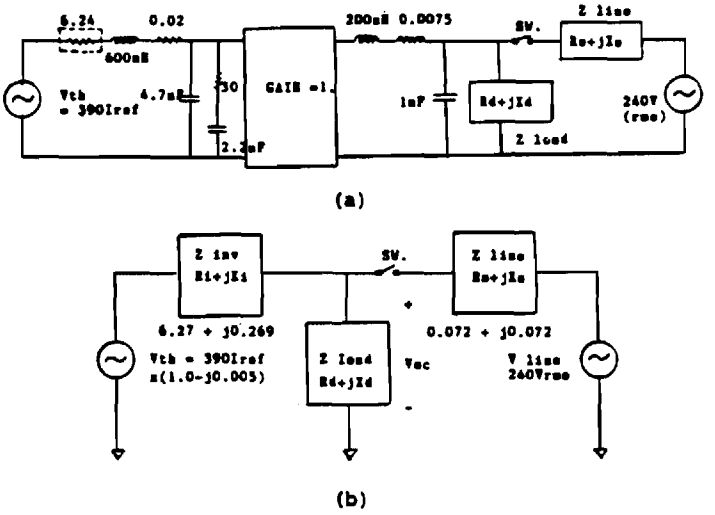


FIGURE II-5 (a) and (b). STEADY STATE EQUIVALENT CIRCUIT OF BLOCK STAGE-UNFOLDER-LOAD-UTILITY LINE.

where Vy is the output of the loop filter. The 1.00 cycle delay program is composed of a delay buffer which has heading and tailing pointers, npnt and dpnt, respectively. The procedure for finding the balanced condition is as follows:

(1) In the simulation program, first, find the steady state solution for Vac before and after the line disconnection.

(2) Second, find the appropriate phase adjust parameters in the PLL circuit subroutine to obtain the longest run-on time when the load is balanced.

(3) Finally, refer to the Purdue simulation model of the Georgia Power experimental data and apply them to the program adjusting the critical point of the Schmitt-trigger threshold level.

The islanding simulation program is written in Turbo-Pascal Version 3.0 on an IBM-PC/AT with the Hercules Monographic Card. In the program modules, the PLL circuit subroutine is constructed via dynamic system state equations and the integrations are performed using a 4^{th} order Runge-Kutta integration subroutine. The buck stage and the unfolder system are implemented via the steady state model associated with the proposed simplified circuit.

The phase adjust model is a one-pole, one-zero system and is included in the PLL circuit,

$$Vac'(s) = \frac{(s + b)}{(s + a)} Vac(s)$$

where a = 420, b = 340. At 60 Hz, the steady state phase shift is 6.04° and provides the phase compensation in the PLL circuit.

II.3 Simulation Results

Figure II-6 is a flow chart of the computer program. The user is requested to input the following program data:

(1) The load impedance, $R_d + jX_d$

(2) The line impedance, $R_s + jX_s$

(3) The level of solar insolation, P_I, in mW/cm^2.

The line voltage is taken to be 240 volts rms. The run-on time may be computed for various values of load impedance and solar insolation in order to obtain a reasonable estimate of major trends which may establish those critical conditions in the PV-utility grid system which maximize the run-on time.

Figure II-7 shows the GPC experimental results. The matched condition is shifted to +10% from the balanced power requirements. Figure II-8 shows the computed simulation results from the simplified model. Although the run-on times are relatively longer than those of Figure II-7, they show that the longest run-on time occurs at the balanced condition. The longest run-on time obtained is 1.168 seconds.

Figure II-9 shows the resulting variations by changing the load vars. It depicts how run-on times are affected when the reactive power changes, i.e., as the load is varied from capacitive to inductive. The longest run-on time occurs at the balanced condition. But, the graph shows that the run-on time for capacitive loads is longer than that for inductive loads.

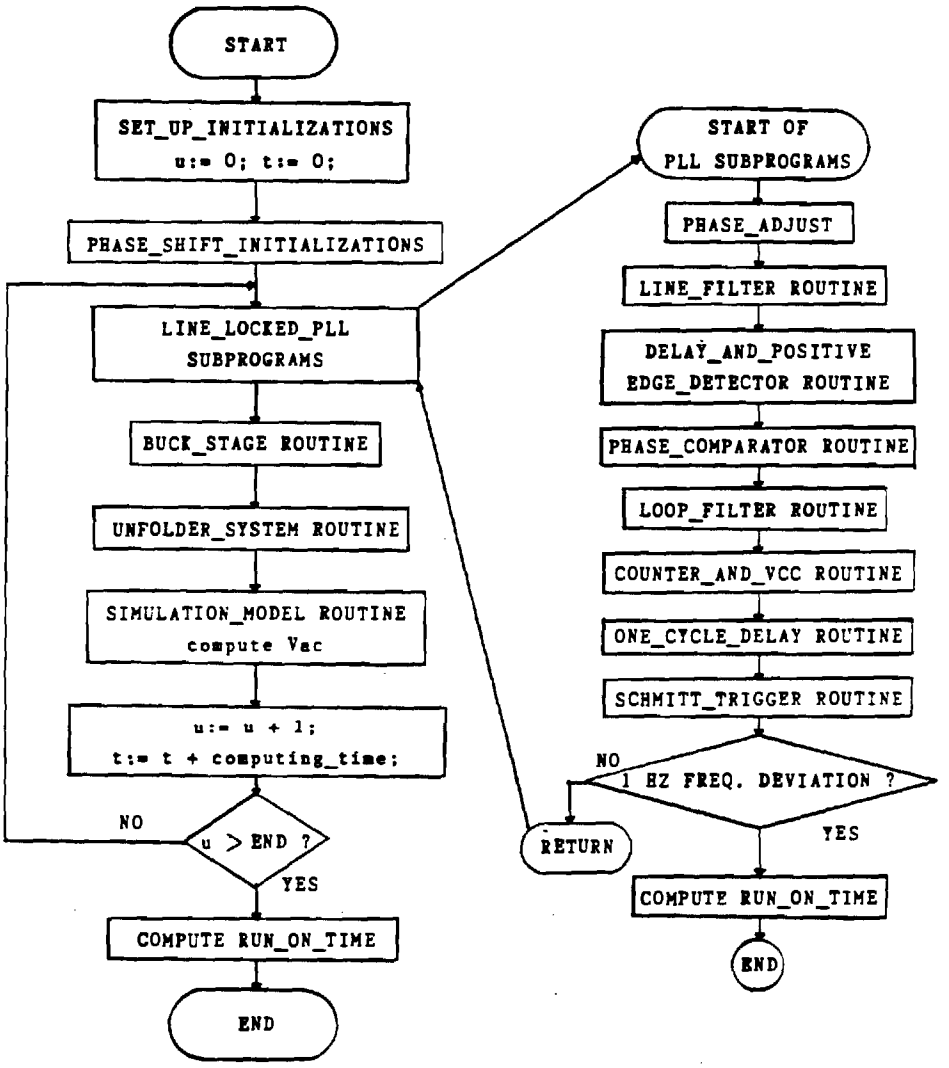


FIGURE II-6. THE FLOW CHART OF THE SIMULATION PROGRAM.

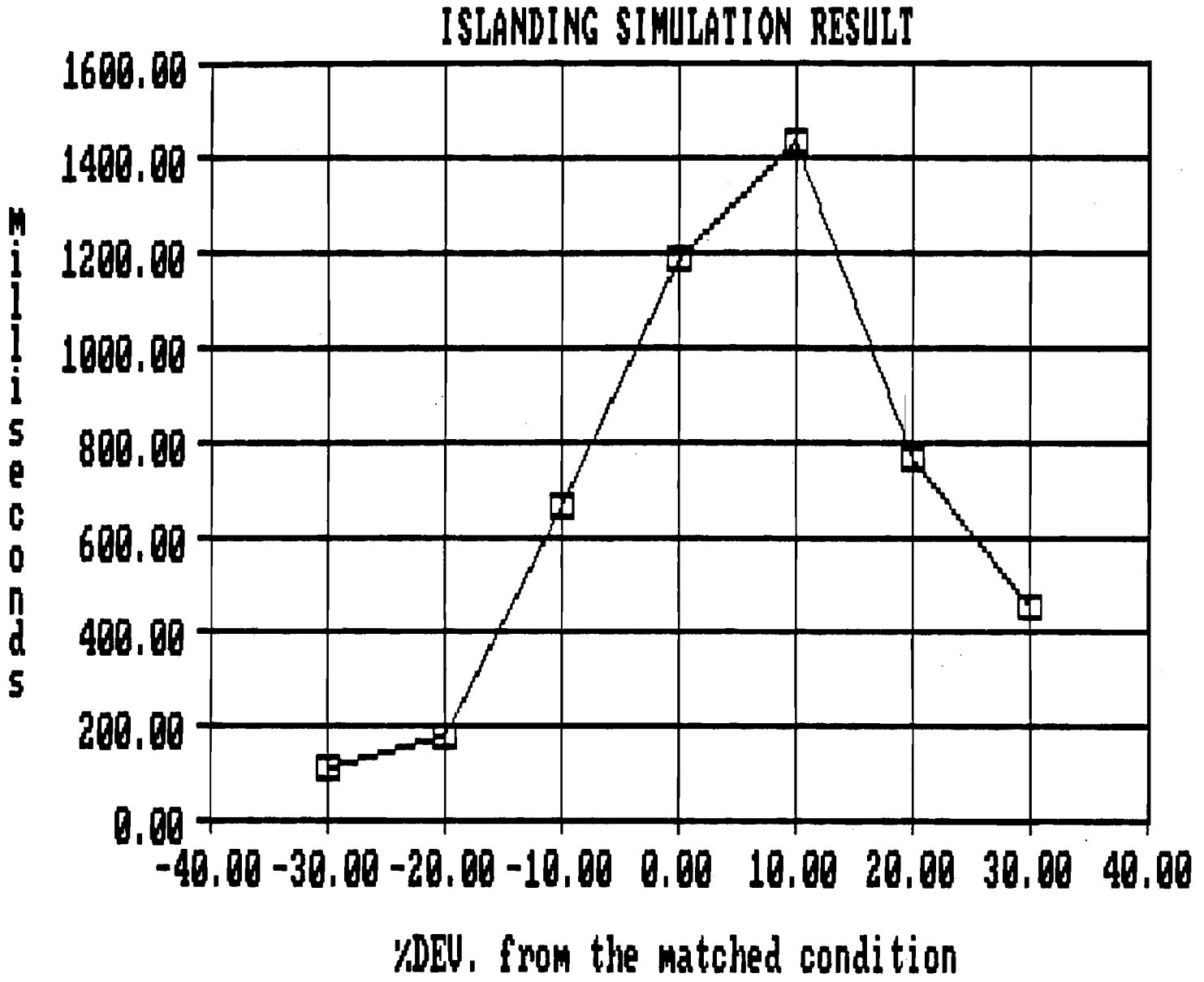


FIGURE II-7. EXPERIMENTAL RESULTS (GEORGIA POWER).

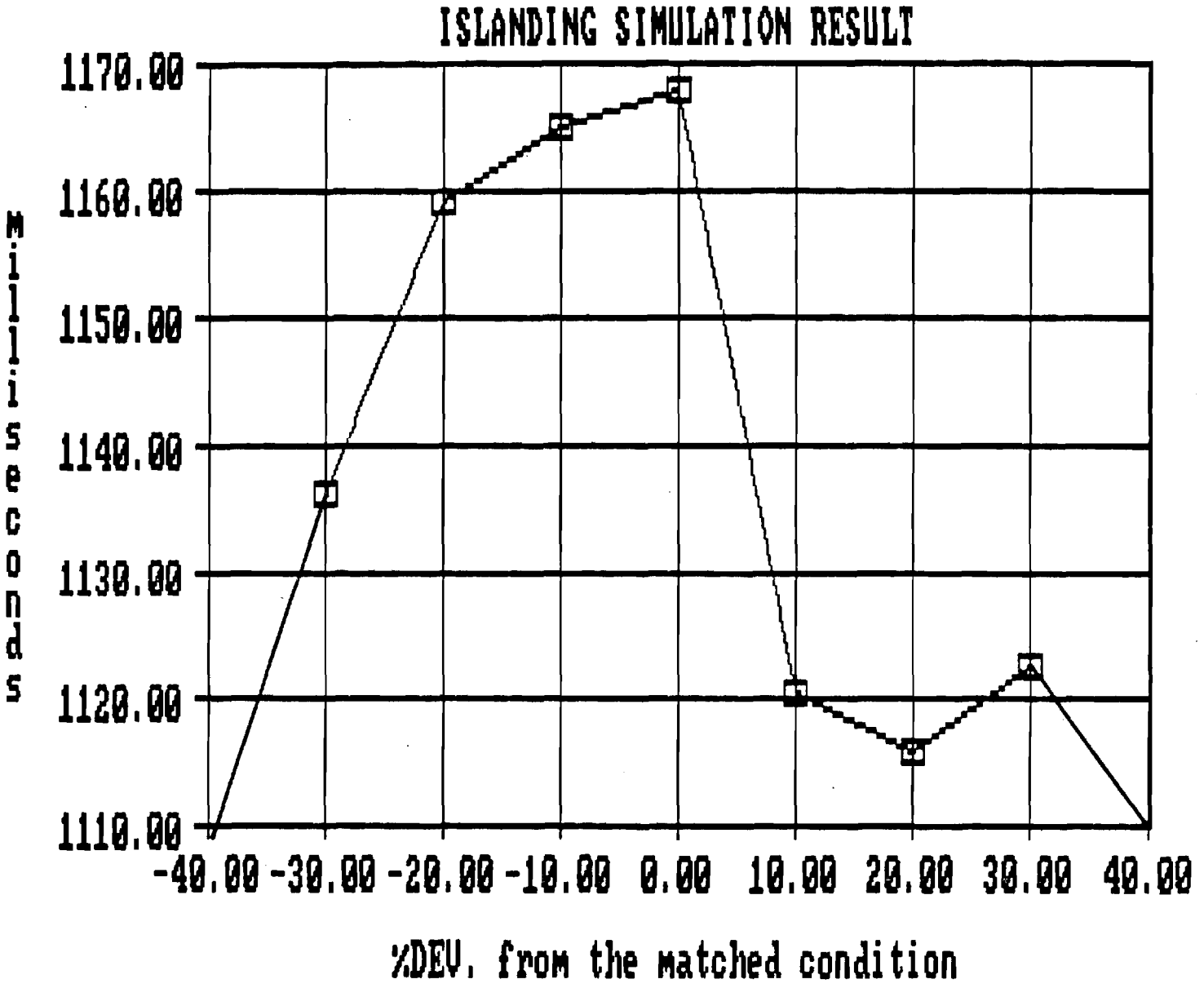


FIGURE II-8. SIMULATION RESULTS (GEORGIA TECH).

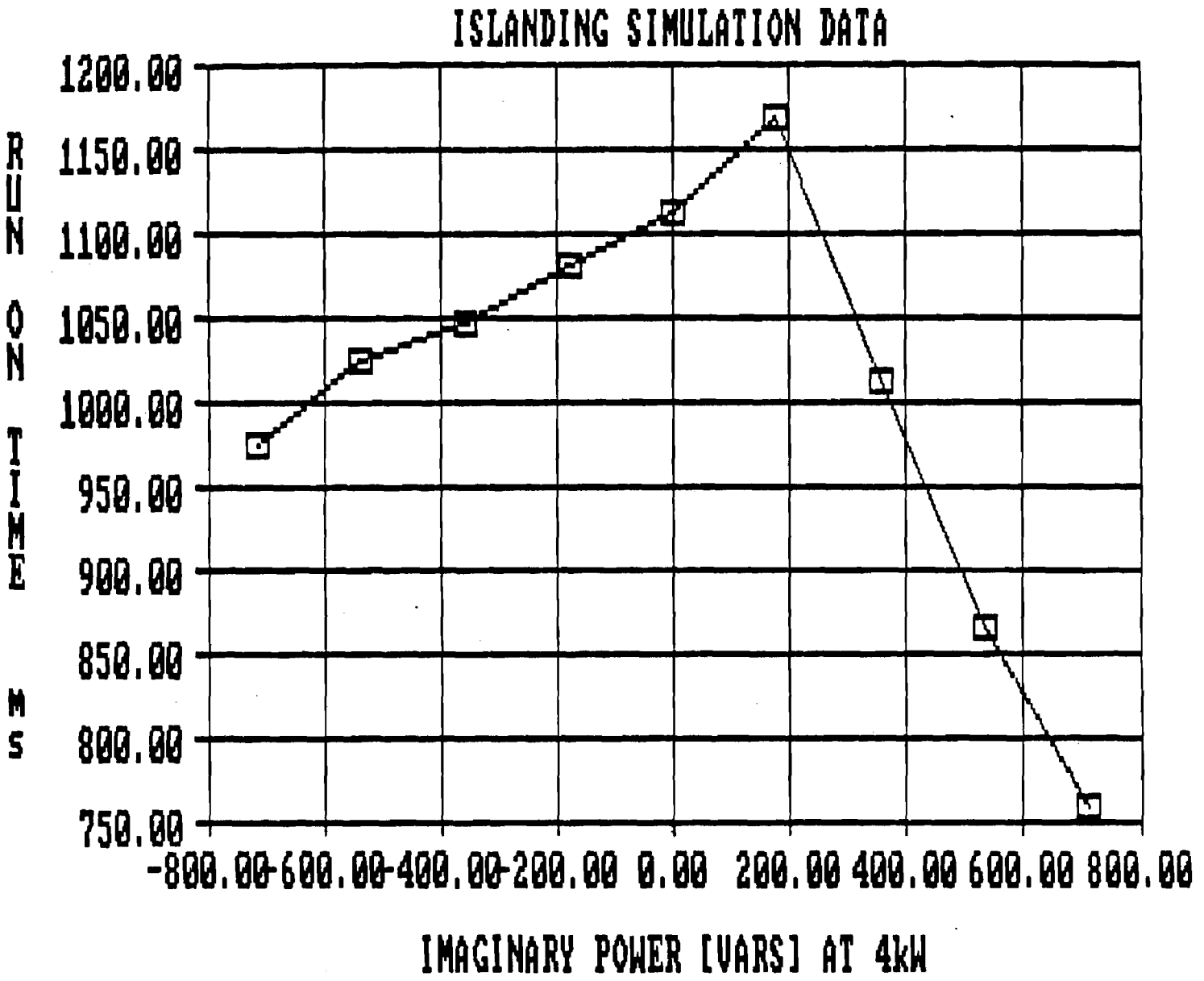


FIGURE II-9. SIMULATION RESULTS (GEORGIA TECH).

II.4 Conclusions

A simplified computer model has been developed for predicting the run-on time of self-commutated inverters operating in a utility interactive mode. The inverter receives DC power from the photovoltaic array and delivers AC power to a local load or the utility lines. The proposed model represents the dynamics of a phase-locked loop control circuitry which is designed to destabilize the inverter operation and shut down the power conditioning subsystem when a phase discrepancy between the line and some reference signal is detected.

The model predicts conservatively the general trend of run-on time variations versus changes in load parameters. Under balanced conditions, maximum run-on time is obtained, as expected. Due to steady state assumptions and component simplifications, the model does not track precisely the variations in run-on times and, therefore, caution should be exercised in its use as a predictive tool.

REFERENCES

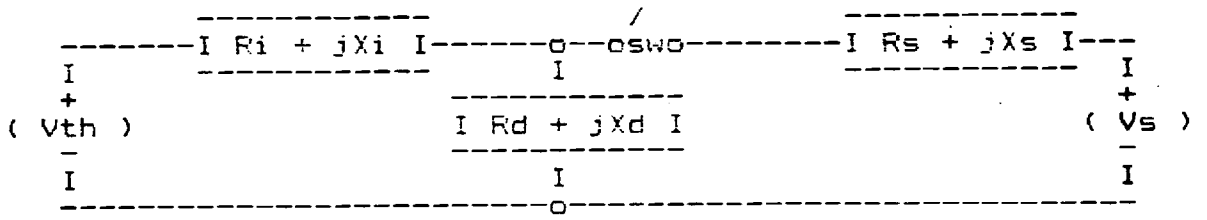
1. A. Cocconi, et al., "High-Frequency Isolated 4 kW Photovoltaic Inverter for Utility Interface," Power Conversion International, May 1984.

2. O. Wasynczuk, et al., "Dynamic Simulation of Dispersed, Grid-Connected, Photovoltaic Power Systems: System Studies," SAND83-7019 Report, March 1985.

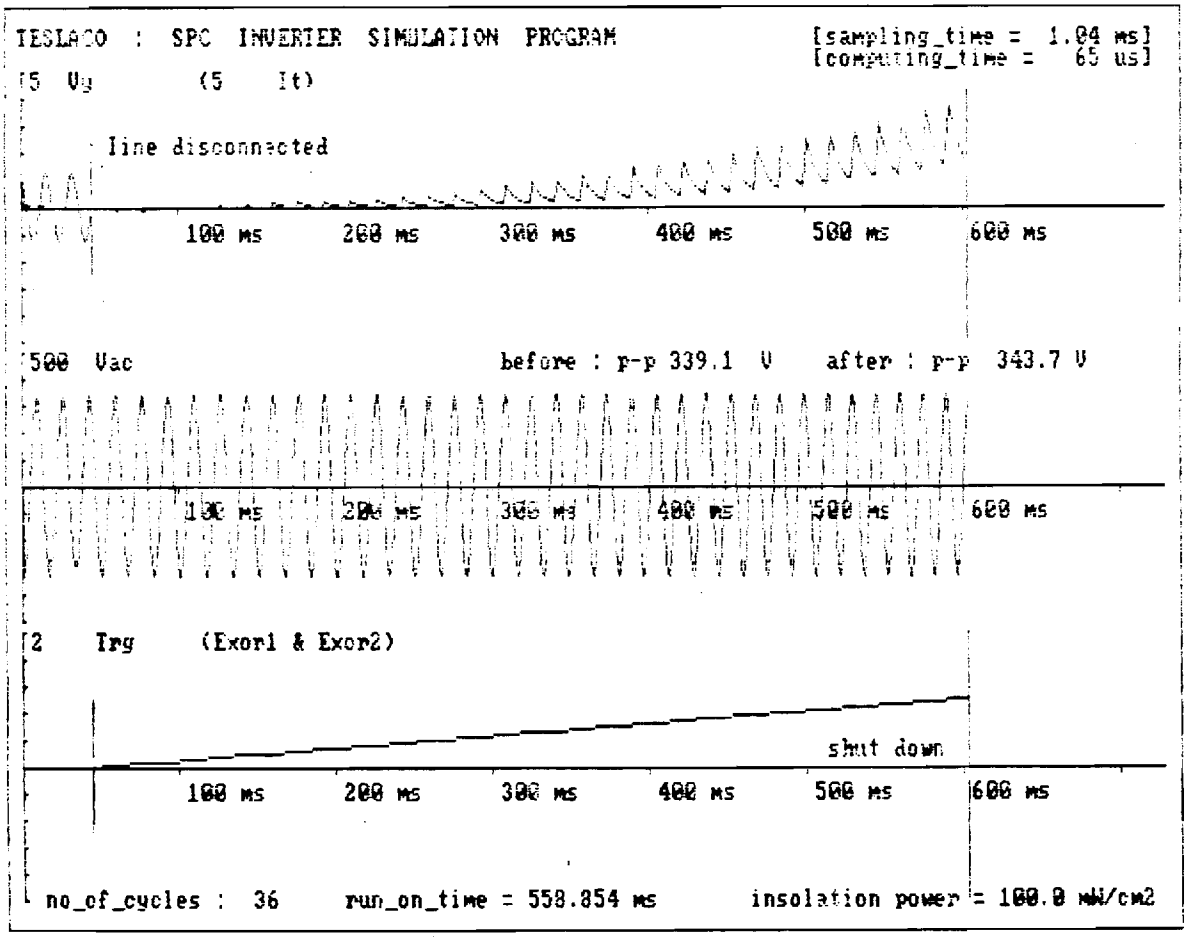
APPENDIX II-1

SIMULATION PROGRAM AND TYPICAL COMPUTER RESULTS

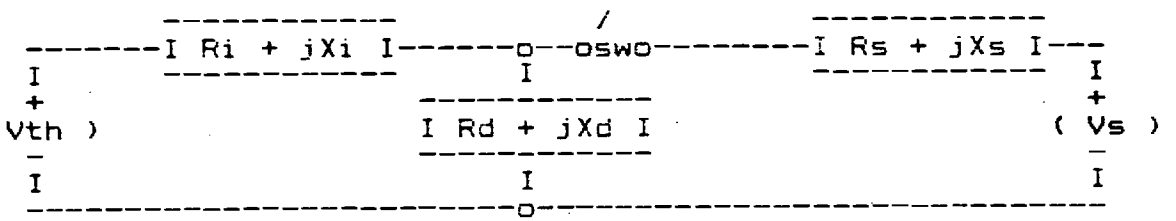
IA1A. Purdue DATA [Highly Inductive Load]




```
Vpva = 199.3 [V]     Pi = 100.0 [mW/cm2]      |Iref|  =    1.1971
Vth      =    390 Iref ( 1.000 + j   0.002)   [volts     ]
Vs       =    339.4                           [volts(p-p)]
Ri + jXi =      6.270     + j     0.280        [ohms]
Rd + jXd =     16.050     + j     5.590        [ohms]
Rs + jXs =      0.072     + j     0.072        [ohms]
Load Power =  3194.65     + j  1112.65         [Watts,Vars]

before line_disconnection =    339.10 (V)       0.087 deg
after  line_disconnection =    343.74 (V)       4.585 deg

PRESS ANY KEY TO CONTINUE ...
```



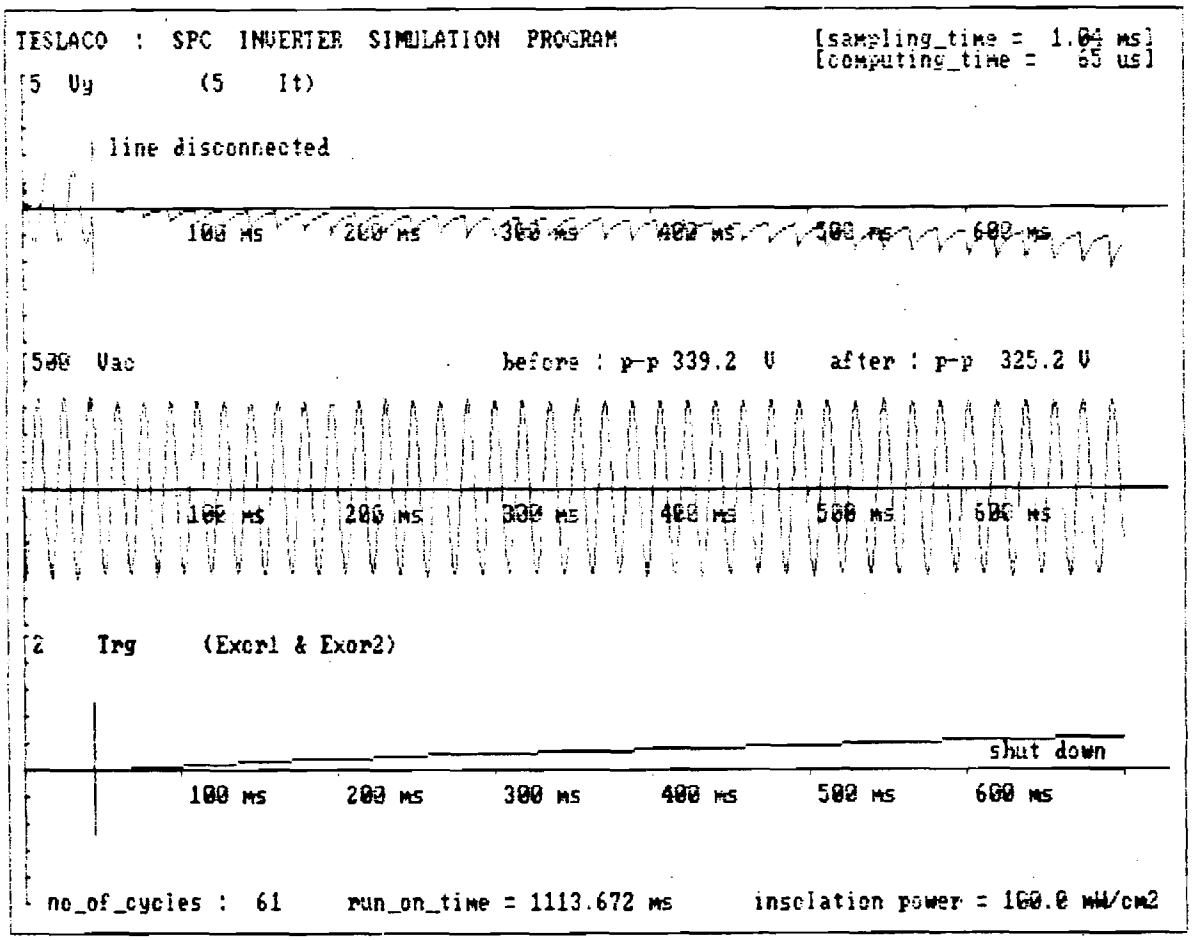
IAIV. 0% [Matched] R_{Load} = 14.4 [ohms] (Resistive Matching)



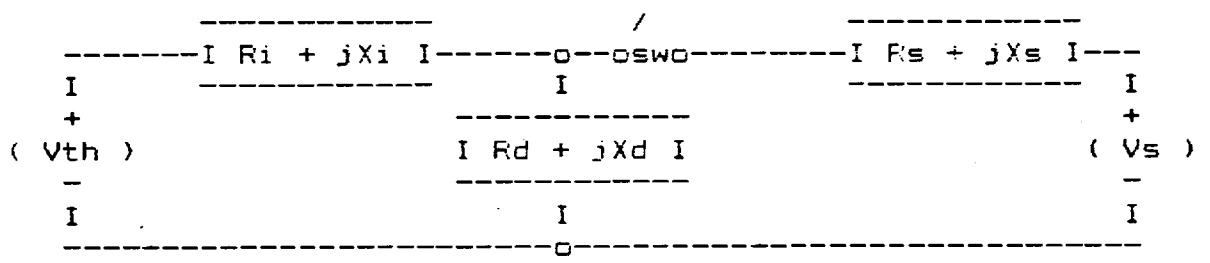
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1971
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 14.400 + j 0.000 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 3995.45 + j 0.00 [Watts,Vars]

before line_disconnection = 339.22 (V) -0.043 deg
after line_disconnection = 325.16 (V) -0.679 deg

PRESS ANY KEY TO CONTINUE ...



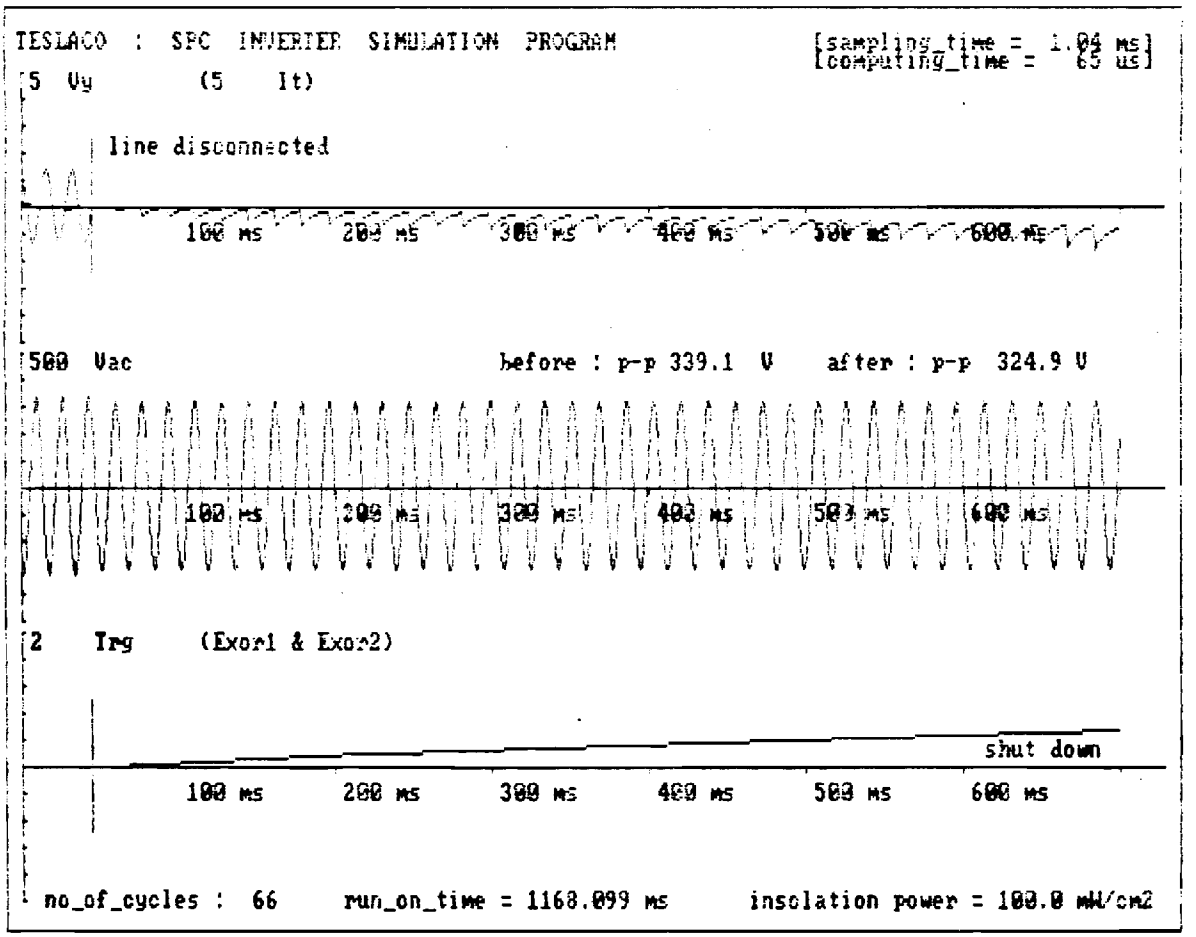
IA1X. 0% [Matched] Z_{Load} = 14.317 + j 2.7286[mH]//5.0[uF]



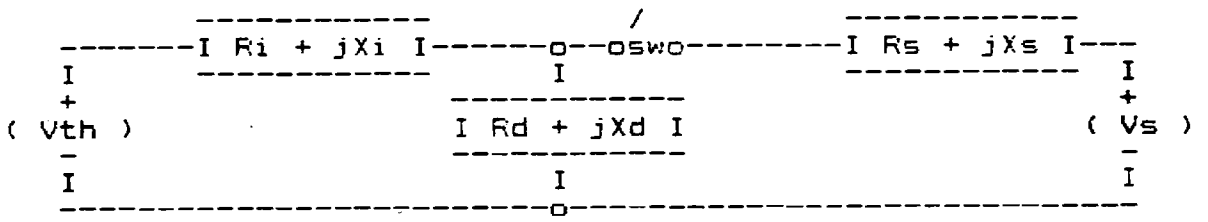
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1971
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 14.362 + j 0.642 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 3996.21 + j 178.72 [Watts,Vars]

before line_disconnection = 339.14 (V) -0.036 deg
after line_disconnection = 324.93 (V) 0.099 deg

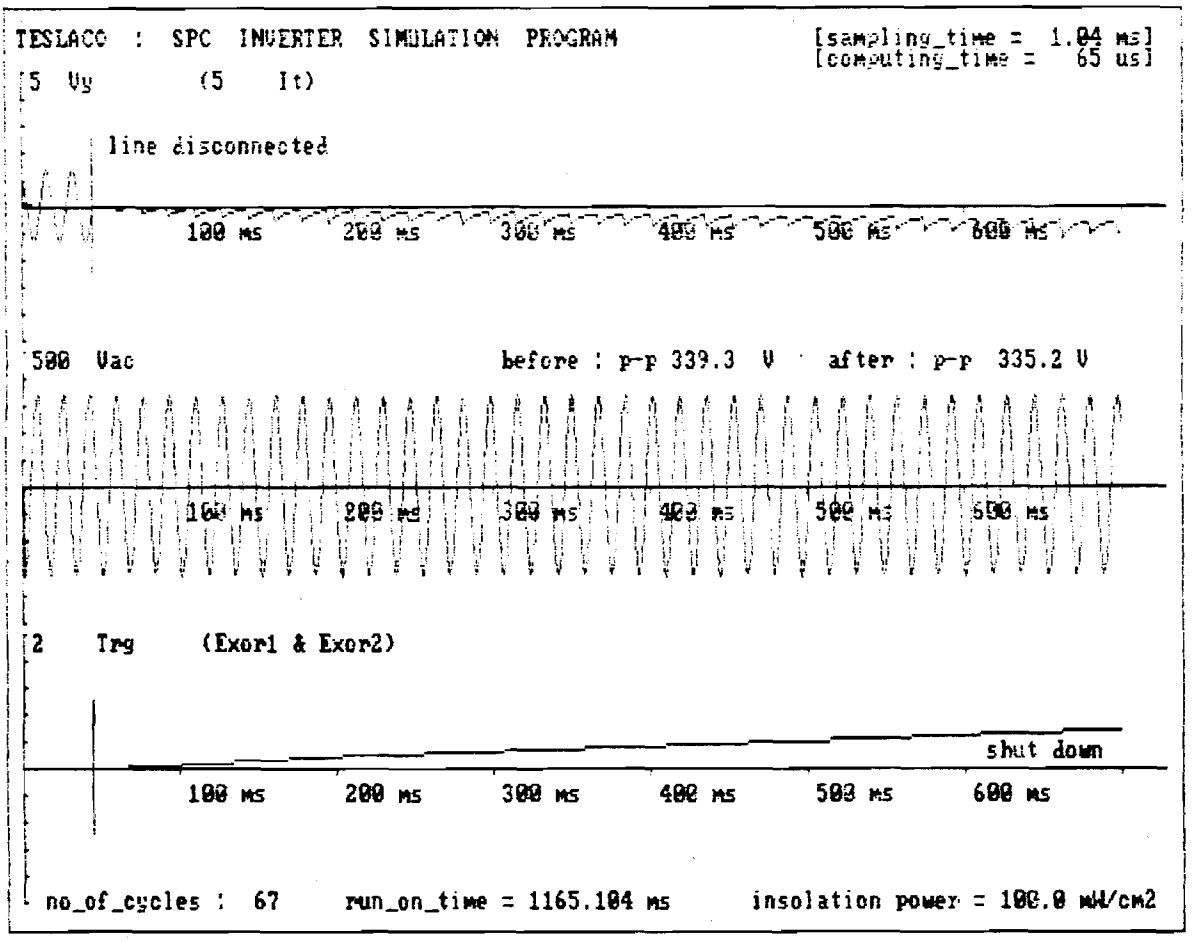
PRESS ANY KEY TO CONTINUE ...



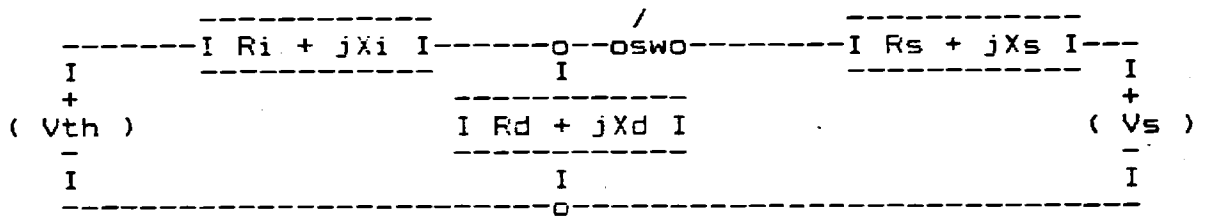
IA2X. -10% Z_{Load} = 15.899 + j 3.3688[mH]//5.0[uF]




```
Vpva = 199.3 [V]     Pi = 100.0 [mW/cm2]     |Iref|    =    1.1971
Vth        =    390 Iref ( 1.000 + j  0.002)  [volts      ]
Vs         =   339.4                          [volts(p-p)]
Ri + jXi   =     6.270     + j     0.280       [ohms]
Rd + jXd   =    15.961     + j     0.794       [ohms]
Rs + jXs   =     0.072     + j     0.072       [ohms]
Load Power =  3597.83      + j   178.88        [Watts,Vars]

before line_disconnection =   339.31 (V)    -0.008 deg
after  line_disconnection =   335.15 (V)     0.179 deg

PRESS ANY KEY TO CONTINUE ...
```



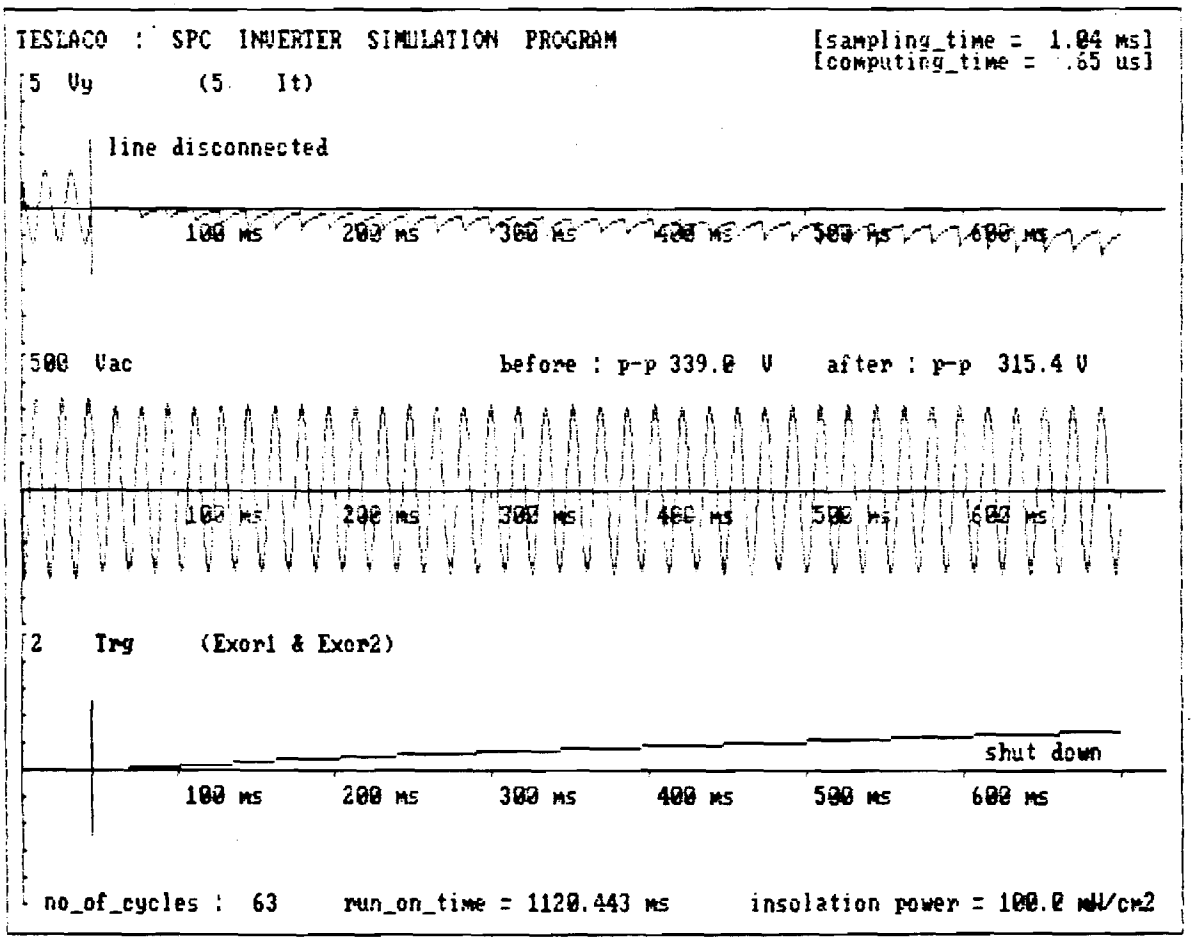
IA3X. +10% Z_{Load} = 13.017 + j 2.5975[mH]//7.0[uF]



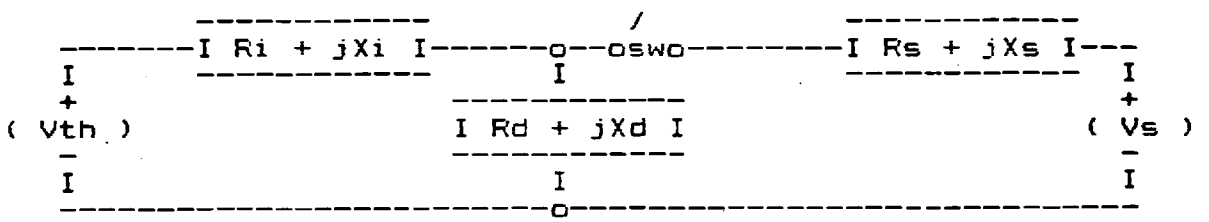
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1971
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 13.069 + j 0.532 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 4388.79 + j 178.54 [Watts,Vars]

before line_disconnection = 338.98 (V) -0.063 deg
after line_disconnection = 315.43 (V) 0.024 deg

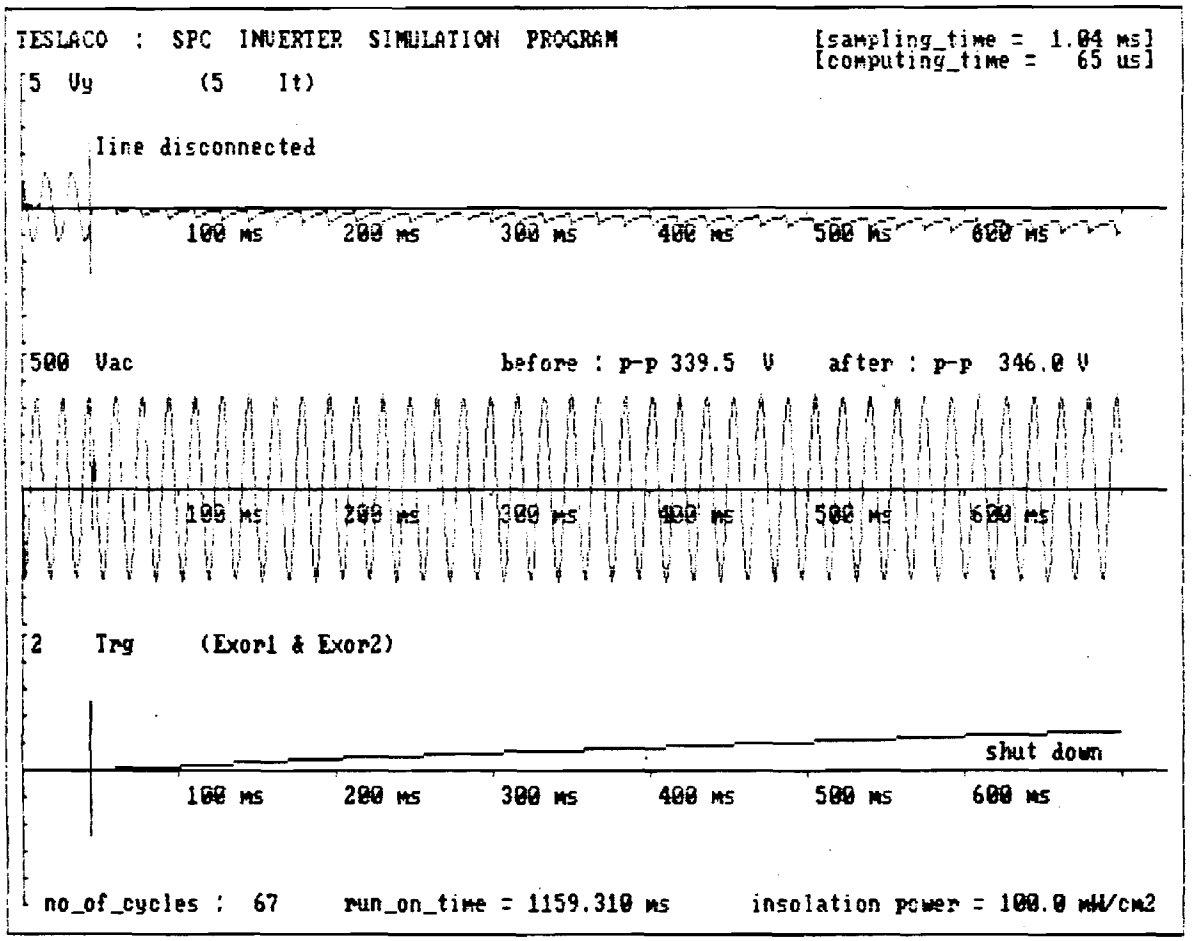
PRESS ANY KEY TO CONTINUE ...



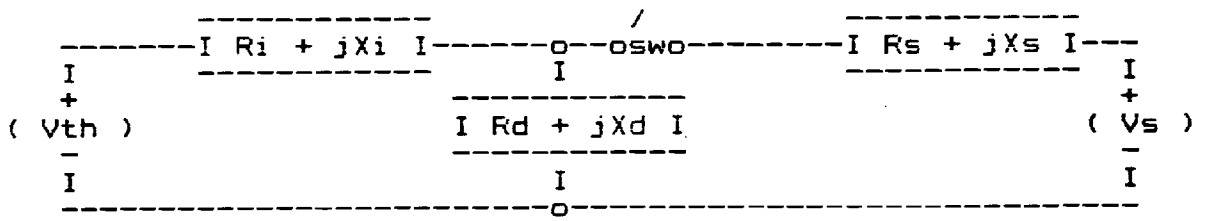
IA4X. -20% Z_{Load} = 17.856 + j 4.2564[mH]//5.0[uF]



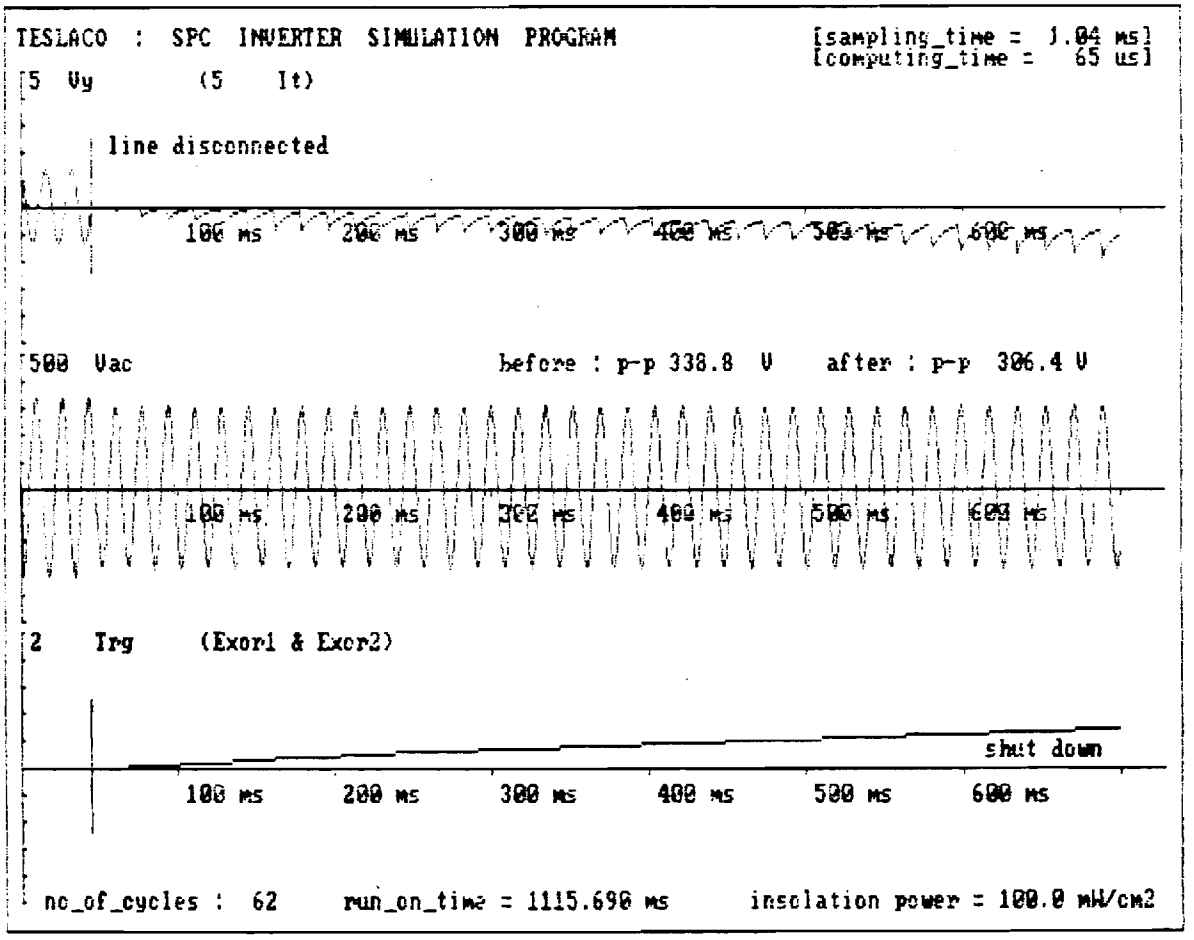

```
Vpva = 199.3 [V]     Pi = 100.0 [mW/cm2]      |Iref|    =    1.1971
Vth       =   390 Iref ( 1.000 + j  0.002)    [volts      ]
Vs        =   339.4                           [volts(p-p)]
Ri + jXi  =     6.270     + j     0.280        [ohms]
Rd + jXd  =    17.944     + j     1.004        [ohms]
Rs + jXs  =     0.072     + j     0.072        [ohms]
Load Power =  3201.32     + j   179.07         [Watts,Vars]

before line_disconnection =    339.48 (V)      0.020 deg
after  line_disconnection =    345.97 (V)      0.264 deg

PRESS ANY KEY TO CONTINUE ...
```



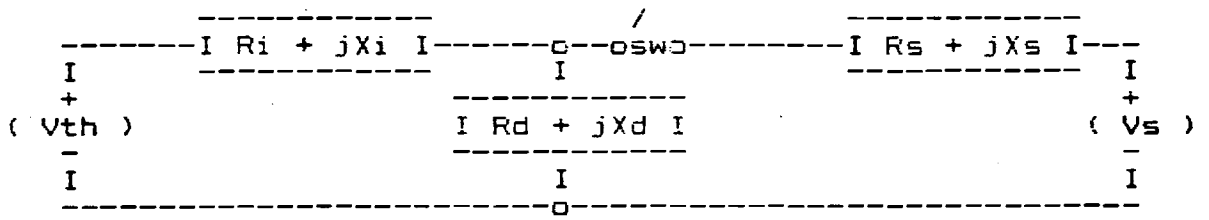
IA5X. +20% Z_{Load} = 11.936 + j 2.3264[mH]//8.0[uF]




```
Vpva = 199.3 [V]     Pi = 100.0 [mW/cm2]     |Iref|    =    1.1971
Vth       =    390 Iref ( 1.000 + j  0.002)  [volts     ]
Vs        =  339.4                           [volts(p-p)]
Ri + jXi  =    6.270    + j    0.280         [ohms]
Rd + jXd  =   11.984    + j    0.447         [ohms]
Rs + jXs  =    0.072    + j    0.072         [ohms]
Load Power =  4782.73   + j  178.33          [Watts,Vars]

before line_disconnection =   338.81 (V)     -0.091 deg
after  line_disconnection =   306.42 (V)     -0.047 deg

PRESS ANY KEY TO CONTINUE ...
```



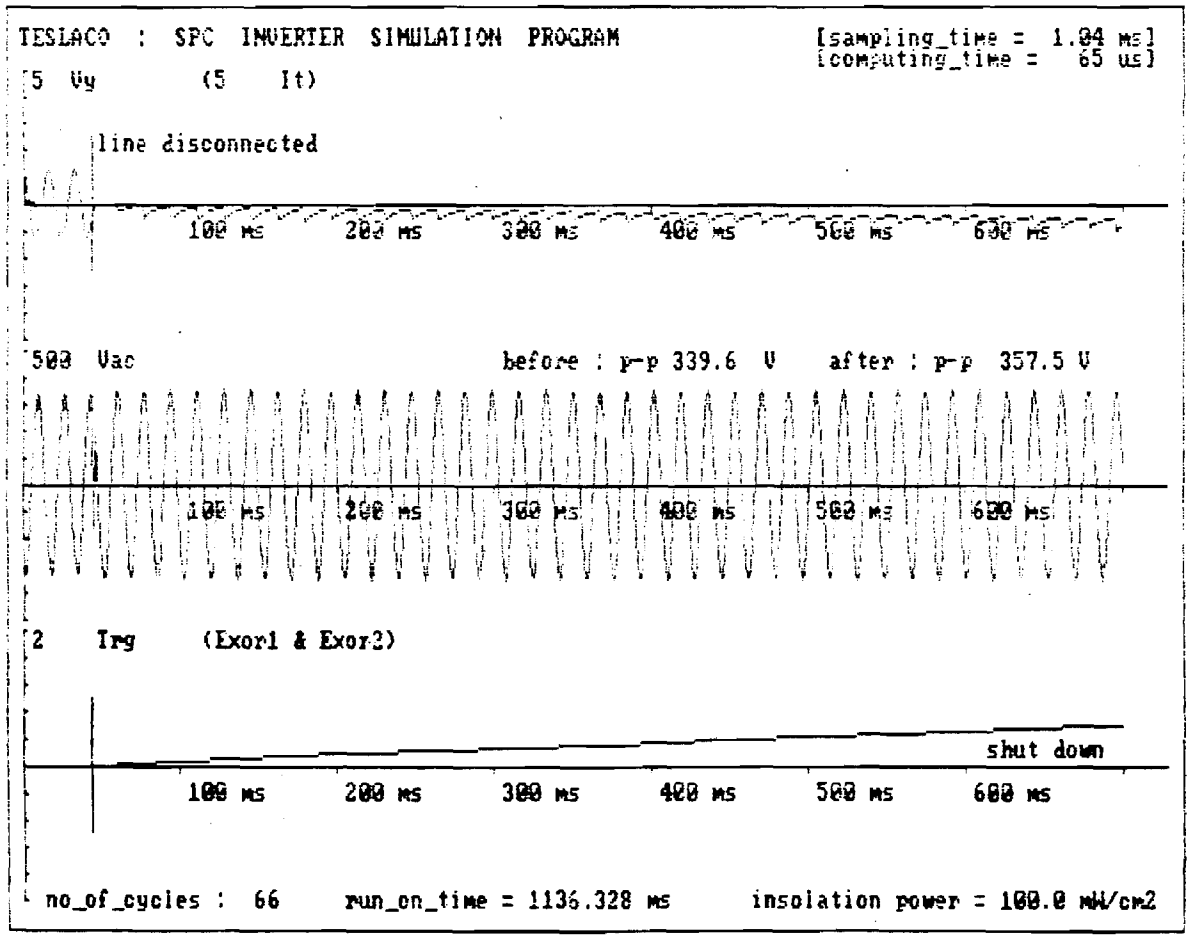
IA6X. -30% Z_{Load} = 2.0357 + j 5.5458[mH]//5.0[uF]



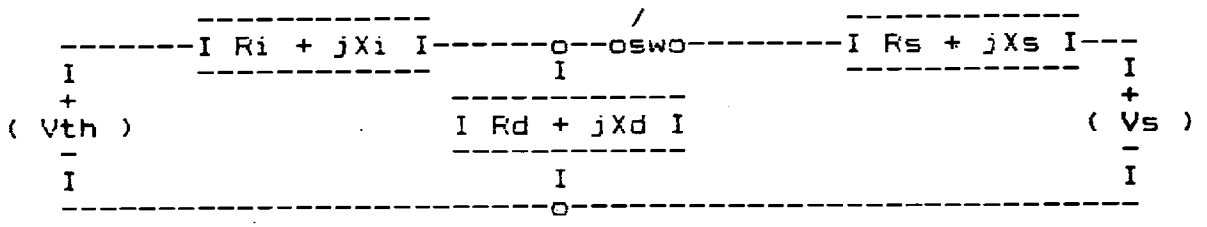
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1971
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 20.488 + j 1.310 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 2803.94 + j 179.24 [Watts,Vars]

before line_disconnection = 339.65 (V) 0.048 deg
after line_disconnection = 357.50 (V) 0.355 deg

PRESS ANY KEY TO CONTINUE ...



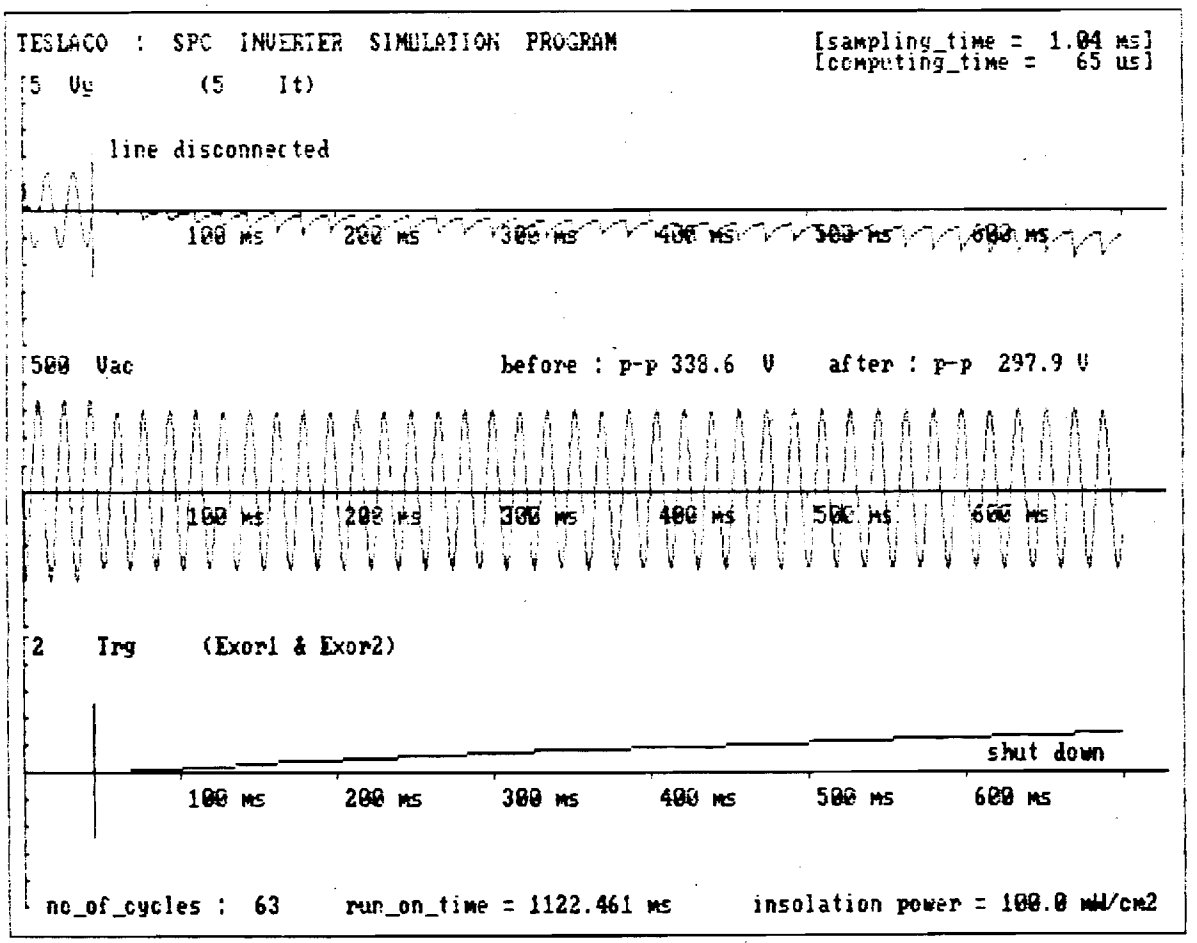
IA7X. +30% Z_{Load} = 11.02 + j 2.105[mH]//9.0[uF]



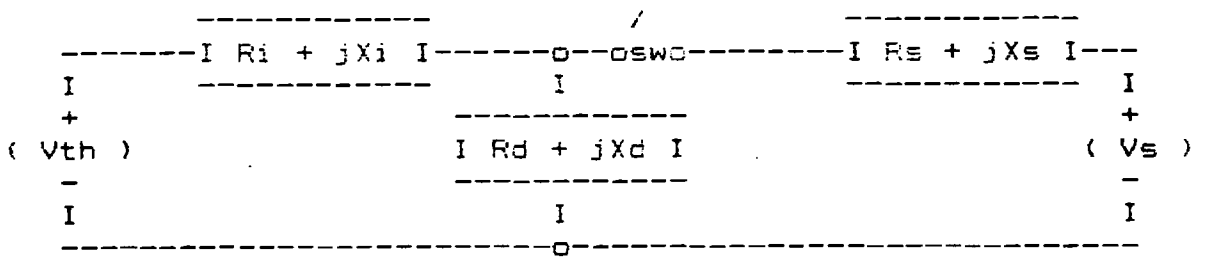
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1971
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 11.064 + j 0.381 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 5176.21 + j 178.20 [Watts,Vars]

before line_disconnection = 338.64 (V) -0.118 deg
after line_disconnection = 297.90 (V) -0.114 deg

PRESS ANY KEY TO CONTINUE ...



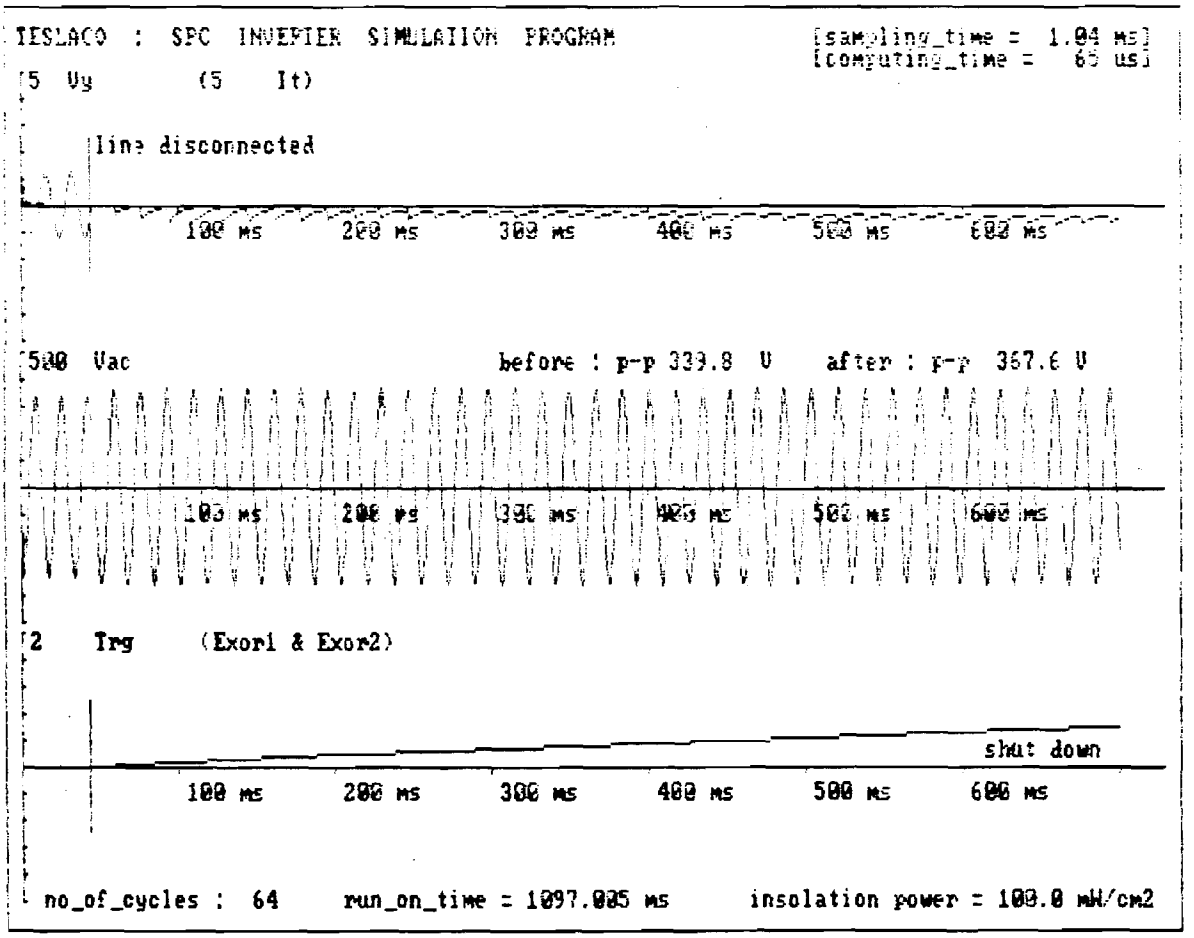
IA8X. -40% Z_{Load} = 23.0 + j 7.1[mH]//5.0[uF]



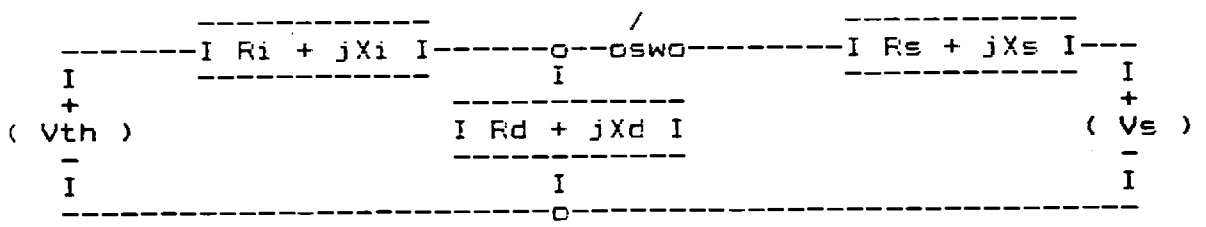
Vpva = 199.3 [V] Pi = 100.0 [mW/cm2] |Iref| = 1.1771
Vth = 390 Iref (1.000 + j 0.002) [volts]
Vs = 339.4 [volts(p-p)]
Ri + jXi = 6.270 + j 0.280 [ohms]
Rd + jXd = 23.190 + j 1.680 [ohms]
Rs + jXs = 0.072 + j 0.072 [ohms]
Load Power = 2476.43 + j 179.38 [Watts,Vars]

before line_disconnection = 339.79 (V) 0.071 deg
after line_disconnection = 367.58 (V) 0.435 deg

PRESS ANY KEY TO CONTINUE ...

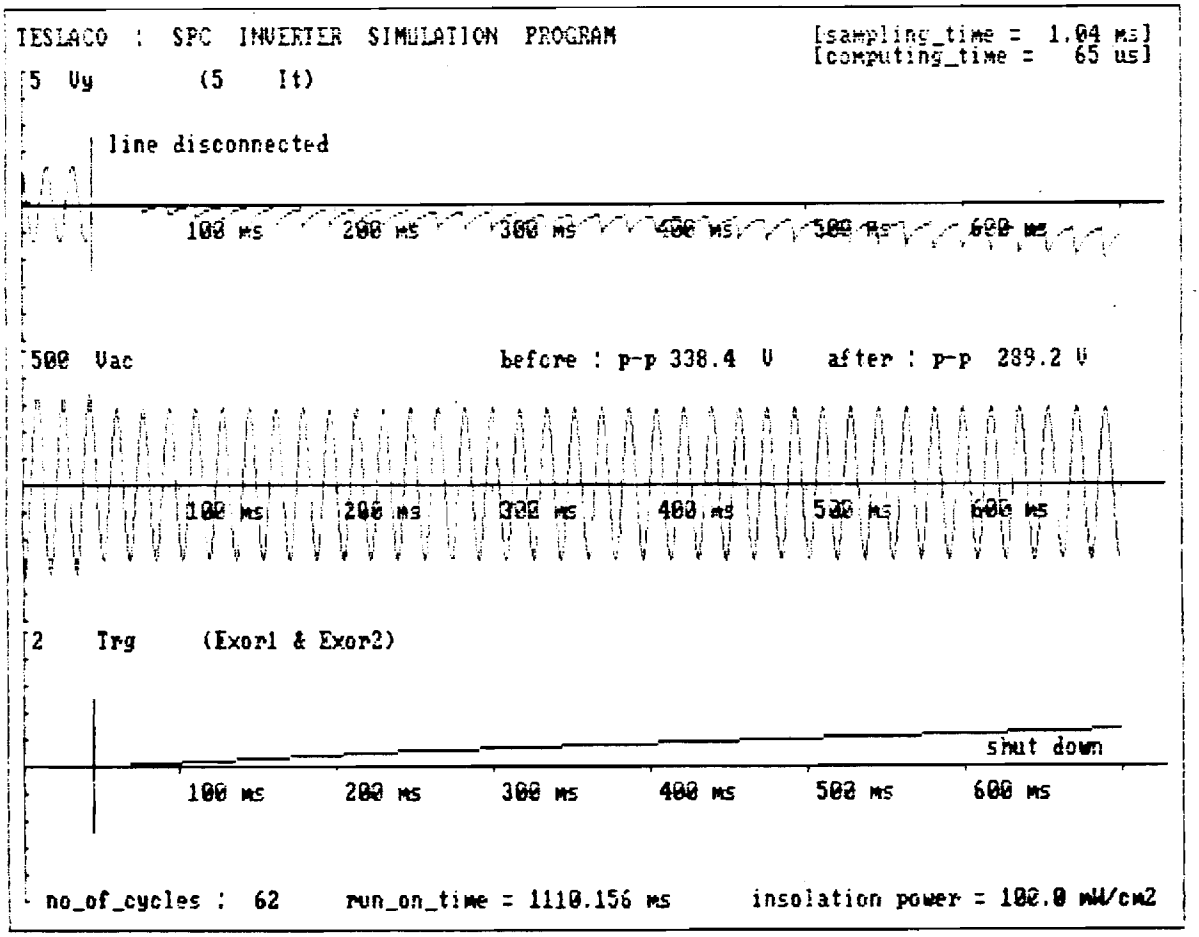


IA9X. +40% Z_{Load} = 10.18 + j 1.9[mH]//10.0[uF]




```
Vpva = 199.3 [V]     Pi = 100.0 [mW/cm2]      |Iref|    =    1.1971
Vth       =    390 Iref ( 1.000 + j  0.002)   [volts      ]
Vs        =   339.4                           [volts(p-p)]
Ri + jXi  =     6.270     + j     0.280        [ohms]
Rd + jXd  =    10.220     + j     0.325        [ohms]
Rs + jXs  =     0.072     + j     0.072        [ohms]
Load Power =  5598.61     + j   178.00         [Watts,Vars]

before line_disconnection =   338.46 (V)     -0.148 deg
after  line_disconnection =   289.25 (V)     -0.182 deg
```

PRESS ANY KEY TO CONTINUE ...



```
{ ***************************************************************************
  *
  *                THIS PROGRAM SIMULATES THE EFL INVERTER (TESLACO)
  *
  *                1987. 1.        Dr. George Vachtsevanos
  *                                S.R.O.      Hoon      Hong
  *
  *                                School of Electrical Engineering
  *                                Georgia Tech.
  *
  ***************************************************************************}

PROGRAM Simulation;

{$I Graphlib.PAS}

CONST pi= 3.14159; dsize= 500; dsize= 500;
      steady_time= 3000;      {make it divided by 16}
      a0= +8.35; a1= 14.67; a2= 3.77;
      aa= 4.6925E6; bb= 4.315; cc= 137.74;
      buck1= 10433.33; buck2= 1666.67;
      ufa= 57.392; ufb= 6.7448E3; ufc= 1.3408E5; uid= 4.5[illegible]
      ufs1= 37.5; ufs2= 5000;
      critical= 1.0;          {fine tuning}

TYPE  arrays= array[1..4] of real;
      delayed= array[0..dsize] of real;

VAR   b,v,e,x,y,v0,e0,x0,y0,v1,e1,x1,y1,v2,e2,x2,y2,
      v3,e3,x3,y3,ino,ipc: arrays;
      buf,ewr: array[0..dsize] of real;
      ex: char; flag,shutdown: boolean;
      vacd,irefc,vsd: delayed;
      ux,u,t,k1,k2,k3,k4,vy,vc1,vc1o,vac,vs,iref,ib,iu,it,its,dd,acc,[illegible]
      vacp,cnt,exoro,w,rt,qt,qvcc,io,imag,b1,itau,check,[illegible]: real;
      dpnt,npnt,rr,vv,start_time,[illegible]: integer;
      hw0,hx0,hy0,hz0,hu0,hs0,ht0,hw1,hx1,hy1,hz1,hu1,hs1,ht1,
      vw0,vx0,vy0,vz0,vu0,vs0,vt0,vw1,vx1,vy1,vz1,vu1,vs1,vt1: integer;
      crt,exor,exori1,exori2: integer;
      vrms,be_icm,be_icms,be_icas,be_icmp,be_icap,af_icm,af_icas: real;
      phase_vs,phase_iref,vacmax,it_mag,it_phase,vsx,irefx,
      alpha,beta,freq,freq_adj,pi_power: real;

procedure set_up_initializations;
var  i: integer;
begin
     ClrScr;
     repeat
          Gotoxy(10,10);
          write('Mono/Color Graphics Card [Mono=0/Color=1] ? ');
          readln(crt);
     until crt in [0..1];
```

```
      ClrScr;
      for i:=1 to 10 do writeln;
      write('                                    JUST A MOMENT ...');
      b[1]:= 1; b[2]:= 2; b[3]:= 2; b[4]:= 1;
      k1:= 1/15360; k2:= 16/15360; x0[1]:= 0; x[1]:= -100;
      x[2]:= 0; x1[1]:= 0; x1[2]:= 0; x2[1]:= 0; x2[2]:= 0; x3[1]:= 0;
      rt:= 0; qt:= 0; apnt:= 0; npnt:= 256; cnt:= 0;
      rr:= 0; vr:= 0; vy:= 0; vzi:= 0; it:= 0; exori:= 0; exori2:= 0;
      for i:=0 to dsize do buf[i]:= 0; exoro:= 0; check:= 0;
      imag:= 1 {i}; pwr:=    ; flag:= false; quitnow:= false;
      ux:= 0; u:= 0; t:= 0; scalefactor:= 20; run_on_time:= 0; sum:= 0;
      w:= 120*pi; { 60 Hz } phase_va:= 0; phase_iref:= 0; vectak:= 0;
      freq:= 920/15360; freq_adj:= 1.2; alpha:= 420; beta:= 340;
      for i:=0 to dsize do
      begin
        vacd[i]:= 339.4*sin(i*k1*120*pi); vsd[i]:= vacd[i];
        irefc[i]:= vacd[i]*imag/339.4;
      end;
      for i:=0 to 256 do cur[i]:= sin(i*pi/180*360/1024);
end; {real_up_initializations}

procedure runge_kutta_4(var e,v,x,y: arrays; order: integer; var t: real);
var  i,j,k: integer;
begin
      case p of 1: for i:= 1 to order do
                   begin
                        e[i]:= k1*v[i]; y[i]:= x[i]; x[i]:= x[i] + 0.5*e[i];
                   end;
                2: for j:= 1 to order do
                   begin
                        x[j]:= k1*v[j]; e[j]:= e[j] + b[p]*x[j];
                        x[j]:= y[j] + b1*x[j];
                   end;
                3: for j:= 1 to order do
                   begin
                        x[j]:= k1*v[j]; e[j]:= e[j] + b[p]*x[j];
                        x[j]:= y[j] + b1*x[j];
                   end;
                4: for k:= 1 to order do
                   begin
                        x[k]:= e[k] + k1*v[k]; x[k]:= y[k] + x[k]/6;
                        exit;
                   end;
      end; {case}
      if p = 2 then b1:= 1;
      if p <> 2 then t:= t + 0.5*k1;
end; {runge_kutta_4}

procedure line_filter;
{ input: vac      output: ino[i] }
var  index,p: integer;
begin
      vacd[round(ux) mod dsize]:= vac;      { phase adjustment }
      b1:= 0.5;
      if u <> 0 then
      begin
           for p:= 1 to 4 do
           begin
```

```
                v2[1]:= (beta-alpha)*vac - alpha*x2[1];
                runge_kutta_4(e2,v2,x2,y2,1,p); { order = 1 }
            end;
            t:= t - k1;
        end;
        vacp:= x2[1] + vac;

        b1:= 0.5;
        if u <> 0 then
        begin
            for p:= 1 to 4 do
            begin
                v[1]:= x[2]+c1*vacp;
                v[2]:= -a1*x[2]-a0*x[1]-c1*a1*vacp;
                runge_kutta_4(e,v,x,y,2,p); { order = 2 }
            end;
            t:= t - k1;
        end;
        x[1]:= x[1];
        if (lno[1]>=0) and (rt<0) then rr:= 1;
        rt:= x[1];
end; {line_filter}

procedure delay_and_positive_edge_detector;
{ output: vv }
begin
        qvcc:= buf[dpnt];
        if (qvcc>=90) and (qt<90) then vv:= 1;
        qt:= qvcc;
end; {delay_and_positive_edge_detector}

procedure phase_comparator;
{ output: io }
begin
        if rr=1 then begin io:= io-0.000227; rr:= 0; end;
        if io<-0.000227 then io:= -0.000227;
        if vv=1 then begin io:= io-0.000227; vv:= 0; end;
        if io<-0.000227 then io:= -0.000227;
end; {phase_comparator}

procedure loop_filter;
{ input: io      output: vy }
var  p: integer;
begin
        b1:= 0.5;
        if u <> 0 then
        begin
            for p:= 1 to 4 do
            begin
                v1[1]:= x1[2]+aa*io; v1[2]:= -cc*x1[2]+aa*(bb-dd)*io;
                runge_kutta_4(e1,v1,x1,y1,2,p); { order = 2 }
            end;
            t:= t - k1;
        end;
        vy:= x1[1];
end; {loop_filter}
```

```
procedure counter_and_vcc;
var  p: integer;
begin
     b1:= 0.5;
     if u <> 0 then
     begin
          for p:= 1 to 4 do
          begin
               x3[1]:= x0;
               runge_kutta_4(e3,x3,x3,y3,1,p); { order = 1 }
          end;
          t:= t - k1;
     end;
     cnt:= x3[1];
     if cnt>=256 then cnt:= 0;
end; {counter_and_vcc}

procedure one_cycle_delay;
begin
     qvcc:= cnt/256*360; buf[inpnt]:= qvcc;
     inpnt:= (inpnt + 1) and bmax; rpnt:= (inpnt + 1) and bmax;
end; {one_cycle_delay}

procedure schmitt_trigger;
var  p,exor11_old: integer;
     st: string5;
begin
     if flag=true then
     begin
          exor11_old:= exor11;
          if ino[1]>=0 then exor11:= 1;
          if ino[1]<0  then exor11:= 0;
          if flag=true then
             if (exor11_old=0) and (exor11=1) then sum:= sum + 1;
          exor12:= 0;
          if (cnt>=64) and (cnt<192) then exor12:= 1;
          exor:= exor11 xor exor12;                    {exclusive_or}
          exoro:= exor*scalefactor;                    {make 2V volts}
          b1:= 0.5;
          if u <> 0 then
          begin
               for p:= 1 to 4 do
               begin
                    v0[1]:= exoro;
                    runge_kutta_4(e0,v0,x0,y0,1,p); { order = 1 }
               end;
               t:= t - k1;
          end;
          check:= x0[1];
     end;
     if (round(ux) mod 16)=0 then
     begin
          Str(vy:7:3,st); hgrwrite(st,300,320,crt);
          Str(check:7:3,st); hgrwrite(st,300,328,crt);
          Str(vacmax:5:1,st);
          if flag=false then hgrwrite(st,400,128,crt)
                        else hgrwrite(st,600,128,crt);
     end;
     if shutdown=false then
```

```
        if (abs(check)>critical) and (u>steady_time+2000) then
        begin
             shutdown:= true;
             run_on_time:= (u - start_time)/15360*1000;
        end;
end; {schmitt_trigger}

procedure phase_locked_loop;
begin
     line_filter;
     delay_and_positive_edge_detector;
     phase_comparator;
     loop_filter;
     counter_and_vco;
     one_cycle_delay;
     schmitt_trigger;
end; {phase_locked_loop}

procedure buck_stage_and_unfolder;
var index: integer;
begin
     index:= trunc(cnt + phase_iref/pi*128*2) mod 256;
     if index<0 then index:= 256 - index;
     irefx:= imag*swr[index];
     iref:= imag*swr[trunc(cnt) mod 256];
     irefd[round(ux) mod dsize]:= iref;
end; {buck_stage}

procedure simulation_model;
var p: integer; temp: real;
begin
     temp:= 120*pi*t;
     vex:= vrms*sin(temp+phase_vs);
     vs:= vrms*sin(temp);
     vsd[round(ux) mod dsize]:= vs;
     if flag=false then
     begin
          p:= round(ux+1*1) mod dsize;
          ito:= it; it:= it_mag*(vacd[p] - vsd[p]);
          if (u>steady_time+500) and (it>=0) and (ito<0) then
          begin
               flag:= true; start_time:= round(ux);
               phase_iref:= af_lca;
               linex(hx1,74-25,hx1,74+25,crt);
               linex(hx1,282-25,hx1,282+25,crt);
               hgrwrite('line disconnected',hx1+3,f8,crt);
               vacmax:=0;
          end;
          vac:= vex*be_lcm/vrms;
     end;
     if flag=true then
     begin
          it:= 0;
          vac:= 390*irefx*af_lcm;
     end;
     if (abs(vac)>vacmax) then vacmax:= abs(vac);
     t:= t + k1;
end; {simulation_model}
```

```
procedure graph_init;
begin
     initgrf(crt);
     gmode(crt);
     clearscr(crt);
     linex(0,0,719,0,crt); linex(719,0,719,343,crt);
     linex(719,343,0,343,crt); linex(0,343,0,0,crt);
     hgrwrite
     ('TESLACO  :  SRC  INVERTER  SIMULATION  PROGRAM  5.2',crt);
     hgrwrite('[sampling_time =  1.04 ms]',490,8,crt);
     hgrwrite('[computing_time = 65.1 us]',490,16,crt);
{    hgrwrite('by  Hoon  Kang    1986.12.',490,328,crt); }
     hgrwrite('insolation power =        mW/cm2',450,328,crt);
     Str(pi_power:5:1,st); hgrwrite(st,600,328,crt);
     hgrwrite('#_of_steps = ',20,328,crt);
     frame('5  Vy          (5     It)',10,74,crt);
     frame('500  Vac                  ',10,176,crt);
     frame('1     Trg      (Ecr=1 : Exc=2)',10,232,crt);
     hgrwrite('before : p-p          V',300,128,crt);
     hgrwrite('after  : p-p          V',300,138,crt);
     hw1:= 10; hv1:= 10; hv1:= 10;  hz1:= 10;  hu1:= 10;  hs1:= 10;
     vw1:= 76-trunc(0.5+hit*10); vx1:= 74-trunc(0.5+vy*10);
     vv1:= 178-trunc(0.5+vac*0.1); vu1:= 312-trunc(0.5+exori2*5);
     vs1:= 282-trunc(0.5+check*50); vz1:= 302-trunc(0.5+exori1*5);
     vt1:= 282-trunc(0.5+cnt*0.04);
end; {graph_init}

procedure graph_result;
begin
     vw0:= vw1; vw1:= 74-trunc(0.5+hit*10); hw0:= hw1; hw1:= hw1 + 5;
     linex(hw0,vw0,hw1,vw1,crt);
     vx0:= vx1; vx1:= 74-trunc(0.5+vy*10); hx0:= hv1; hx1:= hx1+5;
     linex(hx0,vx0,hx1,vx1,crt);
     vy0:= vy1; vy1:= 178-trunc(0.5+vac*0.1); hy0:= hy1; hy1:= hy1+5;
     linex(hy0,vy0,hy1,vy1,crt);
     vs0:= vs1; vs1:= 282-trunc(0.5+check*50); hs0:= hs1; hs1:= hs1+5;
     linex(hs0,vs0,hs1,vs1,crt);
     vu0:= vu1; vu1:= 312-trunc(0.5+exori2*5); hu0:= hu1; hu1:= hu1+5;
     linex(hu0,vu0,hu1,vu1,crt);
     vz0:= vz1; vz1:= 302-trunc(0.5+exori1*5); hz0:= hz1; hz1:= hz1+5;
     linex(hz0,vz0,hz1,vz1,crt);
     vt0:= vt1; vt1:= 282-trunc(0.5+cnt*0.04); ht0:= ht1; ht1:= ht1+5;
     linex(ht0,vt0,ht1,vt1,crt);
end; {graph_result}

procedure phasor_init;
var  ri,xi,rd,xd,rs,xs,vacsr,vacsi,vacpr,vacpi,vacr,vaci,angl,ang2,ang3,
     alfa,beta,denom,tempr,tempi,tempr1,tempi1,pve_power,pt0,vpve,ang: real;
     x,y,z: real;

     procedure complex_mult(x1,x2,y1,y2: real; var z1,z2: real);
     begin
          z1:= x1*y1 - x2*y2;
          z2:= x1*y2 + x2*y1;
     end;
```

```
procedure complex_div(x1,x2,y1,y2: real; var z1,z2: real);
var  temp: real;
begin
     temp:= y1*y1 + y2*y2;
     if temp=0 then begin
                         writeln('divided by zero error');
                         exit;
                    end;
     z1:= x1*y1 + x2*y2; z1:= z1/temp;
     z2:= x2*y1 - x1*y2; z2:= z2/temp;
end;

function arctan1(xi,xr: real): real;
var  arct: real;
begin
     if xr=0 then
        if xi<0 then arct:= -pi/2
           else if xi>0 then arct:= pi/2
                else arct:= 0
     else arct:= Arctan(xi/xr);
     if xr<0 then arct:= arct + pi;
     Arctan1:= arct;
end;

function f(x: real): real;
begin
  f:= 1.7533E-4*x*(exp(x/21.26)-1)-22.12*(x-pi_power/10)*pva_pive
end;

function df(x: real): real;
begin
  df:=1.7533E-4*(1+x/21.26)*exp(x/21.26)-22.12*(pi_power/10)*
end;

procedure find_vpva;
var  xn: real;
begin
     vpva:= 100; xn:= 0;
     while abs(vpva-xn) > 3 do
     begin
          xn:= vpva;
          vpva:= vpva - f(vpva)/df(vpva);
     end;
end;

procedure find_load(x,y,z: real);
begin
     if abs(z)>1E-20 then
     begin
          complex_mult(x,w*y*1E-3,0,-1E6/(w*z),tempr,tempi);
          complex_div(tempr,tempi,x,(w*y*1E-3 - 1E6/(w*z)),rd,xd);
     end;
end; {find_load}

begin
     ClrScr;
     denom:= 15.81*15.81+(4.55E5)*(4.55E5)/(w*w);
     alfa:= (150.99 + (2.07E11)/(w*w))/denom;
```

```
beta:= ((-2.848E6)/w)/denom;
beta:= 0.0017;
ri:= 5261.3 + (1.296E12)/(w*w))/denom + 0.0075;
xi:= ((1.064E8)/w)/denom + w*(2.0E-4);
xi:= 0.280;
rs:= 0.072; xs:= 0.072; rd:= 16.05; xd:= 5.09; vrms:= 240;
writeln('input your line impedance (Rs+jXs) ');
write  ('       default = 0.072 + j 0.072 [ohm] :  ');
readln (rs,xs);
x:= 14.317; y:= 2.7236; z:= 5;
writeln('input your load impedance (R  L[mH]  C[uF]) : '); readln (x,y,z);
find_load(x,y,z);
writeln('input your load impedance (Rd+jXd)');
write  ('       default = 16.05 + j 5.59  [ohm] :  ');
readln (rd,xd);
writeln('input your line voltage Vs in RMS ');
write  ('       default = 240.0             [rms] :  ');
readln (vrms);
vrms:= vrms*sqrt(2); pi_power:= 100; pva_power:= 4000; pt0:= 0;
writeln('input your insolation Pi (mW/cm^2)');
write  ('       default = 100.0        [mW/cm^2] :  ');
readln (pi_power);
writeln('input your photovoltaic array Pcva');
write  ('       default = 4000                [W] :  ');
readln (pva_power);
find_vpva;
imag:= 0.2*(pt0 + vpva/50);

ClrScr;
writeln;
writeln('              ------------                                  ');
writeln('    -------I Ri + jXi I------o--csw--------I Rs + jXs I---- ');
writeln('    I         ------------       I                         I ');
writeln('    +                        ------------                  ');
writeln('    Vth                      I Rd + jXd I                  ');
writeln('    -                        ------------                  ');
writeln('    I                             I                        I ');
writeln('    ---------------------------o-------------------------- ');
writeln; writeln;
writeln('    Vpva = ',vpva:5:1,' [V]     Pi = ',pi_power:5:1,' [mW/cm^2]
         '    Iref    =    ',imag:6:4);
writeln('    Vth      =    390 Iref (',
         alfa:6:3,' + j ',beta:6:3,')  [volts      ]');
writeln('    Vs       =   ',vrms:6:1,
                                                 '[volts(p-p)]');
writeln('    Ri + jXi =   ',ri:8:3,'     + j ',xi:8:3,'         [ohms]  ;
writeln('    Rd + jXd =   ',rd:8:3,'     + j ',xd:8:3,'         [ohms]  ;
writeln('    Rs + jXs =   ',rs:8:3,'     + j ',xs:8:3,'         [ohms]  ;

{ phase shift after line disconnected }
complex_mult(rd,xd,alfa,beta,tempr,tempi);
complex_div(tempr,tempi,ri+rd,xi+xd,vacr,vaci);

{ phase shift before line disconnected : from vs }
complex_mult(ri+rd,xi+xd,rs,xs,tempr,tempi);
complex_mult(ri,xi,rd,xd,tempr1,tempi1);
tempr:= tempr + tempr1; tempi:= tempi + tempi1;
```

```
    complex_div(tempr1,tempi1,tempr,tempi,vacsr,vacsi);

    { phase shift before line disconnected : from 390*iref }
    complex_mult(rs+rd,xs+xd,ri,xi,tempr,tempi);
    complex_mult(rs,xs,rd,xd,tempr1,tempi1);
    tempr:= tempr - tempr1; tempi:= tempi + tempi1;
    complex_div(tempr1,tempi1,tempr,tempi,vacpr,vacpi);
    complex_mult(vacpr,vacpi,alfa,beta,tempr,tempi);
    vacpr:= tempr; vacpi:= tempi;

    { calculate from the above results }
    be_lcms:= Sqrt(vacsr*vacsr+vacsi*vacsi);
    be_lcas:= arctan1(vacsi,vacsr);
    be_lcmp:= Sqrt(vacpr*vacpr+vacpi*vacpi);
    be_lcap:= arctan1(vacpi,vacpr);
    af_lcm := Sqrt(vacr*vacr+vaci*vaci);
    af_lca := arctan1(vaci,vacr);
    complex_div(1,1,rs,xs,tempr,tempi);
    tempr1:= arctan1(tempi,tempr);
    {                                                    { phase adjustment }
    {_neg:= 1/Sqrt(Sqr(temp)+Sqr(tenor );
    phase_lcap:= be_lcap; phase_vs:= be_lcas;
    t:= trunc(ar_lcm/(1*w)+0.5); ang3:= ar_lca+180/pi*  { phase adjustment }
    t:= trunc(0+pi/180*(-18)  + 1.0);                   { phase adjustment }
    arg:= arctan1(xd,rd);
    tempr:= 339.4*be_lcms*cos(be_lcas)+390*imag*ia_lcmp*cos(be_lcap-
    tempi:= 339.4*be_lcms*sin(be_lcas)+390*imag*be_lcmp*sin(be_lcap-
    be_lcm:= Sqrt(Sqr(tempi)+Sqr(tempr));
    arg1:= arctan1(tempi,tempr); ang2:= arg1*180/pi;
    t:= trunc(arg1/ 01*w)+0.0);                         { phase shift }
    tenor:= 0.5*Sqrt(be_lcm)/Sqrt(rd*rd+xd*xd);

    writeln('    Load Power = ',tempr*cos(arg):8:2
                + j ',tempr*sin(arg):8:2,'       {watts,vars }
    write(' ,
    writeln('   before line_disconnection = ',
                be_lcm:8:2,' (V)    ',ang2:7:3,' deg');
    writeln('   after  line_disconnection = ',
                390*imag*af_lcm:8:2,' (V)    ',ang3:7:3,' deg');
    writeln; writeln;
    write  ('    PRESS ANY KEY TO CONTINUE ... '); readln;
end; {phasor_init}


{ main program }


begin
     set_up_initializations;
     phasor_init;
     graph_init;
     while (u<steady_time+70000.0) and (ex<>'q') do
     begin
          phase_locked_loop;
          buck_stage_and_unfolder;
          simulation_model;
```

```
        if (ux>=steady_time) and (ux<steady_time+10752) and
           (round(ux) mod 16 = 0) then graph_result;
        u:= u + 1.00; ux:= ux + 1.00; if ux>=16385 THEN ux:= 1;
        Str(u:5:0,st); hgrwrite(st,120,328,crt);
        if shutdown=true then
        begin
             if ux<steady_time+10752 then
             begin
                  line(hx1,282-50,hx1,282+50,crt);
                  line(hx1,74-50,hx1,74+50,crt);
             end;
             hgrwrite('shut down',hx1-88,248,crt);
             ex:= 'q';
        end;
        if keypressed then read(kbd,ex);
   end;
   hgrwrite('no_of_cycles : ',20,328,crt);
   Str(sum:1,st); hgrwrite(st,170,328,crt);
   Str(run_on_time:7:3,st); hgrwrite('run_on_time = '+st+' ms',20,344,crt);
   readln;
   end.{ main program }
```